COMPUTER AIDS FOR PLA DESIGN

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

SUBRATA GOSWAMI

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

FEBRUARY, 1987

EE-1987-M-GOS-COM.

CERTIFICATE

This is to certify that the work entitled 'COMPUTER AIDS FOR PLA DESIGN' by Goswami, S under my supervision has not been submitted elsewhere for a degree.

M Hasan

Dr. M.M. Hasan
Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur-208 016

ABSTRACT

With the advent of VLSI, designers are increasingly taking help of automation and regular structures for design of complicated Integrated Circuits. Both automation and regular structure can reduce design time appreaciably. Also design of regular structure is amenable to automation. Programmable Logic Array (PLA) is such a regular structure used for implementing combinational logic functions. But all regular structures provide the ease in design at the price of area. Large amount of area is required for implementing regular structures. A sizeable portion of this area is used only for interconnections. Two methods used for better area utilisation in PLAs are combinational logic minimization and folding. Logic minimization reduces the vertical dimension of a PLA and folding can reduce both the vertical and horizontal dimension of a PLA.

Here six aids are developed for design of PLA. They consist of a combinational logic minimizer, a column folder, two stick diagram generators and two layout diagram generators. The minimization program is based on a heuristic algorithm. Th folding procedure is based on Graph Theoy. The layout diagram generators break up the PLA into several modules and arrange them according to the input information. All of these aids ar written in PASCAL.

Some examples are shown. Improvements are also suggested.

ACKNOWLEDGEMENT

I take this opportunity to thank Dr. M.M. Hasan for his inspiring guidance and support throughout this work and also for introducing me to the field of Large Scale Integration.

My heartful thanks to Dr. S.G. Dhande for allowing me to use the CAD Centre facilities.

I am grateful to N. Ravi Shankar, K.V.S. Rayudu, P. Bhattacharya, M. Kumar, A.S. Mandal and D. Dutta for the many exciting moments I spent with them.

Finally, I thank Mr. Yogendra for the pains he took in typing of the thesis.

IIT, Kanpur
February, 1987.

Subrata Goswami

INDEX

		Page
CHAPTER 1	INTRODUCTION	1
CHAPTER 2	CIRCUITS FOR PLA	6
	2.1 Technologies Available	6
	2.2 NMOS, with Enhancement Type Driver and Depletion Type Load	6
	2.3 CMOS Inverter	7
	2.4 Pseudo NMOS	10
	2.5 The Basic Gates	11
	2.6 Comparison	17
CHAPTER 3	DESIGN METHODOLOGY OF THE PLA	19
	3.1 Introduction	19
	3.2 PLA Design	19
	3.3 PLAGE	21
CHAPTER 4	MINIMIZATION	23
	4.1 The necessity of Logic Minimization	23
	4.2 Methods Used for Minimization	23
	4.3 Definition of Terms	23
	4.4 Description of the Minimization Method	24
CHAPTER 5	FOLDING	32
	5.1 Introduction	37
	5.2 Definition of Terms	37
	5.3 Method Used for Folding	40
	5.4 Graph Theoretic Basis of the Folding Problem	40
	5.5 Procedure for Finding the Path	43
	5.6 Procedure	44

		Page
CHPATER 6	STICK DIAGRAM	50
	6.1 Introduction	50
	6.2 For Unfolded PLA	50
	6.3 Folded PLA	55
CHAPTER 7	LAYOUT	56
	7.1 Introduction	56
	7.2 Colour Convention	56
	7.3 Layout Design Rules	56
	7.4 For Unfolded PLA	57
	7.5 Folded PLA	59
CHAPTER 8	EXAMPLES	68
	8.1 Binary to Gray Code Converter	68
	8.2 Seven Segment Display	72
	8.3 3-bit Adder	74
	8.4 Decade Counter with Display	74
	8.5 Conclusion	80
APPENDIX		
A	Users' Manual for PLAGE	81
B	SPICE Simulation	86
REFERENCES		

CHAPTER 1

INTRODUCTION

Since its first appearance in the sixties Integrated Circuits has undergone tremendous changes. More and more devices have been packed into a single chip. The growth in the number of devices in a chip has been exponential[1]. But the concentration of a large number of devices into a chip also brought new problems with it. One is the time required for designing these circuits. With large number of devices (>=100,000) this can become prohibitively long and ~~is~~ totally unacceptable for VLSI circuits. To overcome these difficulties designers are taking resort to automation and regular structures [1,2,3]. Regular structure has the added attraction of reduction of error. Example of such regular structures are Gate Arrays, Programmable Logic Arrays, and standard cells. With the advent of VLSI they have become indispensible tools for designers. But all of them have the disadvantage of taking larger area. Hence they are modified in such a way that the area is optimized.

Here the design of one such device is considered. It is called programmable logic array (PLA). It was first suggested by Fleisher in 1975 [3]. It can be viewed as an off shoot of Read Only Memories (ROMs). It is used for implementing two level combinational logic function.

A ROM can be divided mainly into two parts. One part is called the decoder and the other part is called the memory. The decoder decodes the address provided at the input in the sen that it activates a line called word-line. The wordline activates the required bits of the memory.

As ROMS, PLAs can also be divided into two parts. One part does same thing as decoder but with different intention. It is called the AND plane. Here the word line is called product term. Each product term stands for the product part of the sum of product expression of a boolean function. The other part is called OR plane. This part forms the required sum from the products. They are shown in Fig. 1.1 and 1.2.

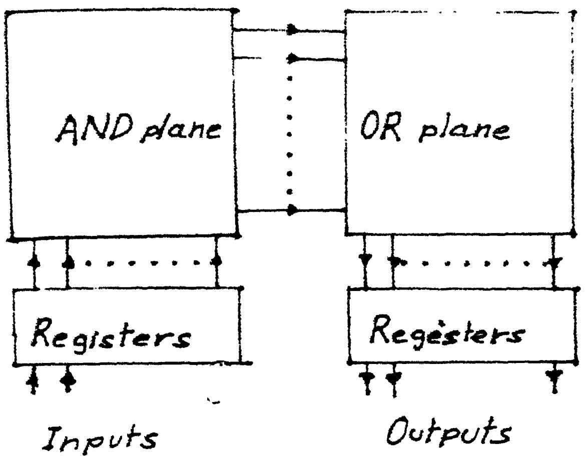
PLAs can be conveniently described by a matrix called personality matrix. For each input and output there is one column in this matrix. Each row of this matrix correspondence to a product term. At the intersection of an input column and a row an I is placed if the non-inverted value of that input is present in that product term. An O is placed if the inverted i value is present. An X is placed if the input is not present in that product term. Similarly an I is placed in a output column if that product term is present in that output. An X is placed if it is not present.

A PLA can be implemented both with bipolar devices and with MOS devices. With bipolar devices it is implemented in AND-OR configuration. In MOS technology it is implemented in NOR-NOR configuration. Here only MOS implementation is described. Circuit diagram of the PLA in Fig. 1.2 is shown in Fig. 1.3.

A great deal of elements in the personality matrix are Xs. This implies that a good deal of silicon area is used for interconnections only. This waste is impermissible. Also to reduce silicon area number of products should be made as small a possible. This is done through combinational logic minimization.The first defect is overcome with folding.

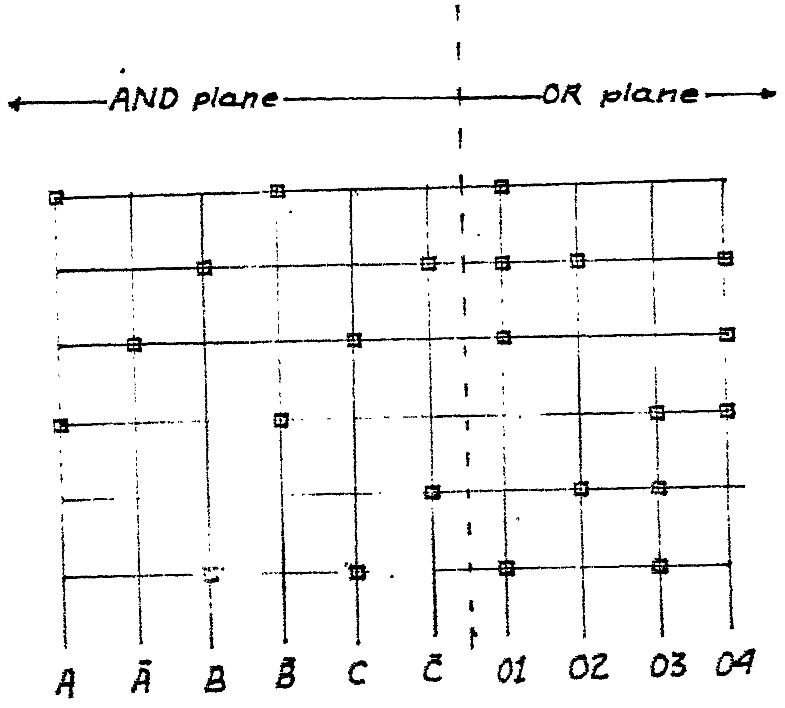
PLAs are used in control circuits of microprocessors. It is used in Intel 8086 and Bit-Mac-32 processors. It is also used in ALUs. Schmookler [4] reported making adders with PLA. Golden [5] reported making complete processers with PLAs. It is also used for the combinational logic in the feedback path of sequential circuits, as multiplier as character generators for cathode ray tube [6].

Chapter 2 deals with circuits used to implement PLAs. Chapter 3 describes the general design approach taken by designers. Chapter 4 and Chapter 5 deal with minimization and folding respectively. Chapter 6 and chapter 7 describe stick diagram generation and layout diagram generation respectively. In chapter 8 some examples are shown.



Genral Structure of a PLA

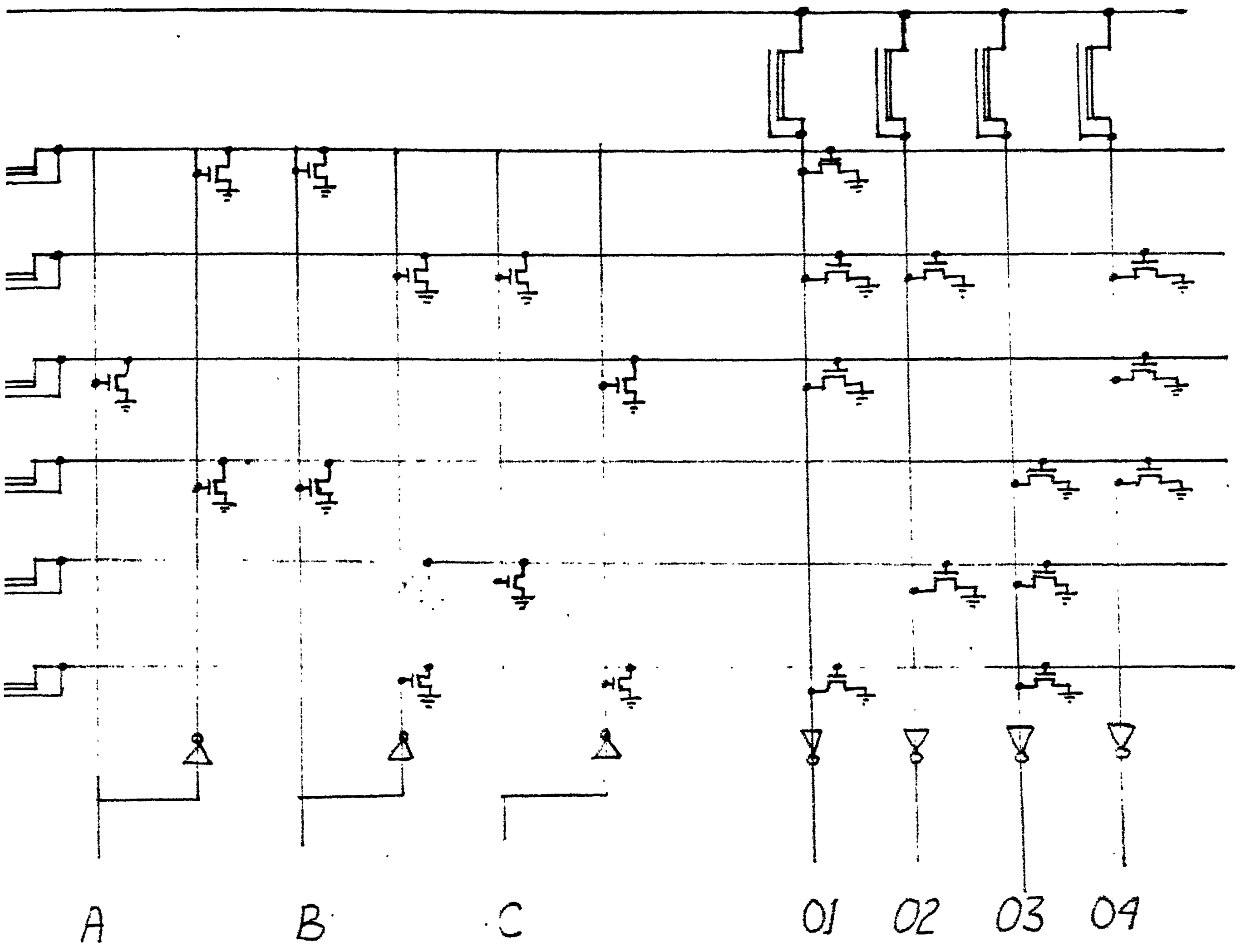
Fig. 1.1



IOXIXXX
XIOIIXI
OXIIXXI
1OXXXII
XXOXIIX
XIIIXIX

Personality Matrix

Fig. 1.2



Circuit Diagram of the PLA in Fig. 1.1
(NMOS Realisation)

Fig.1.3

CHAPTER 2

CIRCUITS FOR PLA

2.1 THE TECHNOLOGIES AVAILABLE

One important consideration in the design of PLAs is the type of device to be used. The most prevalent are the NMOS devices with enhancement type driver and depletion type load. Enhancement type transistor is rarely used as load device. Another type of device is CMOS. There is one more type of device, it is a kind of cross between CMOS and NMOS. It uses a NMOS driver and a PMOS load. The following is an account of them. Only inverters are considered, as they are the basic building block of the above technologies. All the anlyses are carried out through SPICE.

2.2 NMOS, WITH ENHANCEMENT TYPE DRIVER AND DEPLETION TYPE LOAD

Fig. 2.1(a) shows the circuit diagram of such an inverter. Fig. 2.1(b) shows a layout diagram. These type of devices consist the bulk of digital ICs. The following is a SPICE input deck for such an inverter. The device parameters are taken from [20].

```
 NMOS ENH PULLDOWN WITH NMOS DEP LOAD
MPU 3 2 2 0 NDEP L=10U  W=5U AS=100P AD=100P PS=30U PD=30U
MPD 2 1 0 0 NENH L=5U  W=10U AS=100P AD=100P PS=30U PD=30U
C1 2 0 1PF
VSUP 3 0 5V
VIN 1 0 PULSE 0 5 20N 0 0 200N 600N
.DC VIN 0 5 0.25
.PRINT DC V(2) V(1)
.TRAN 10NS 600NS
```

```
.MODEL NENH NMOS LEVEL=3 RSH=35 TOX=300E-10 LD=0.21E-6
+XJ=0.3E-6 VMAX=15E4 ETA=0.18 NSUB=3.5E14 UO=700
+THETA=0.095 VTO=0.78 CGSO=2.8E-10 CGDO=2.8E-10
+CJ=5.75E-5 CJSW=2.48E-10 PB=0.7 MJ=0.5 MJSW=0.3 NFS=1E10
.MODEL NDEP NMOS LEVEL=3 RSH=35 TOX=300E-10 LD=0.21E-6
+XJ=0.3E-6 VMAX=15E4 ETA=0.18 NSUB=35E14 UD=700 THETA=0.035
+VTO=-2.23 CGSO=2.8E-10 CGDO=2.8E-10 CJ=5.75E-5
+CJSW=2.48E-10 PB=0.7 MJ=0.5 MJSW=0.3 NFS=1E10
END
```

The DC transfer characteristic is shown in Fig. 2.4. The transient response is shown in Fig. 2.5.

2.3 CMOS INVERTER

Designers have been trying to use CMOS in PLAs because of its lower power dissipation. Their main disadvantage is that they tend to occupy larger area because every NMOS device is accompanied by a complementary PMOS device. Although PMOSes are inherently slower than NMOSes, CMOS can be made fast at the expense of area. Fig. 2.2(a) shows the circuit diagram and Fig. 2.2(b) shows the layout of a CMOS inverter. The following is a SPICE input deck for such an inverter. The device parameters are taken from [20].

```
 NMOS PULLDOWN WITH PMOS LOAD WITH GATE GROUNDED
MPD 2 1 0 0 NM L=5U W=10U PD=40U PS=40U AS=100P AD=100P
MPU 2 1 3 3 PM L=5U W=10U PD=40U PS=40U AD=100P AS=100P
VSUP 3 0 5V
VIN 1 0 PULSE 0 5 20N 0 0 200N 600N
C1 2 0 1 PF
.DC VIN 0 5 0.25
.PRINT DC V(1) V(2)
.TRAN 10N 500NS
.PRINT TRAN V(1) V(2) I(VSUP)
```

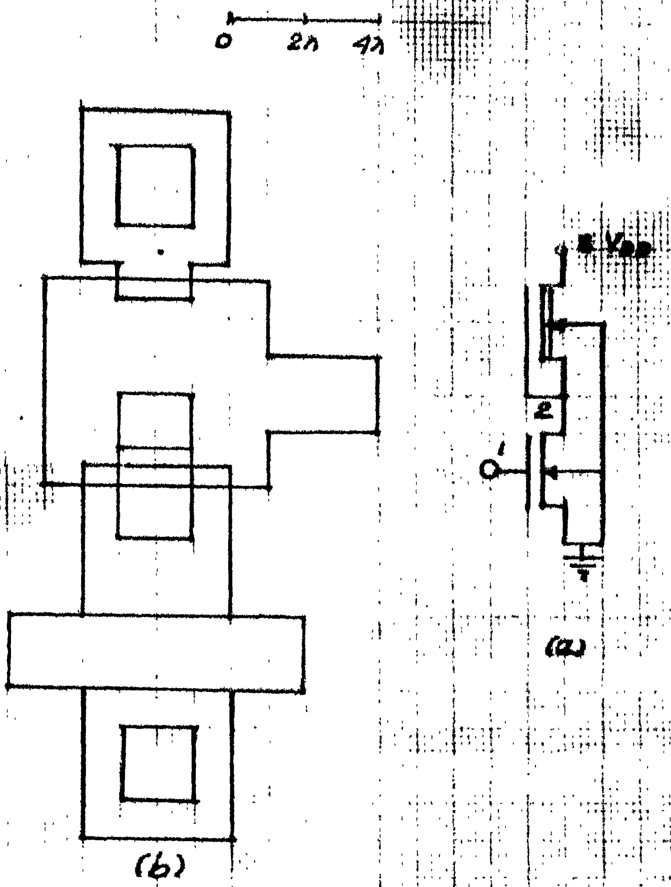
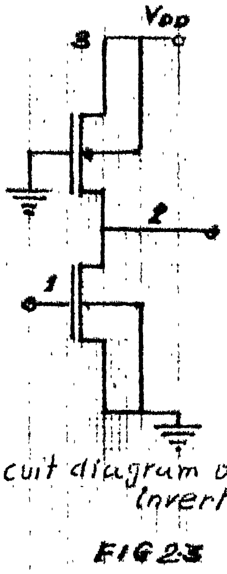
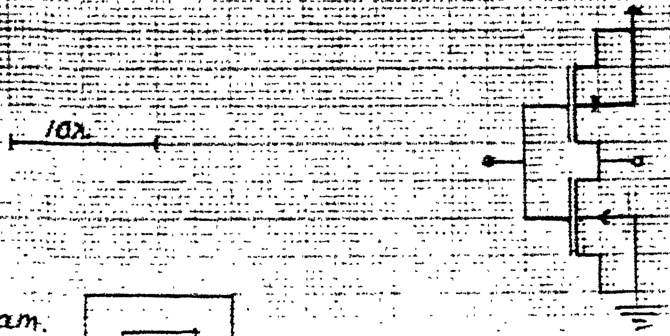


FIG 2.1

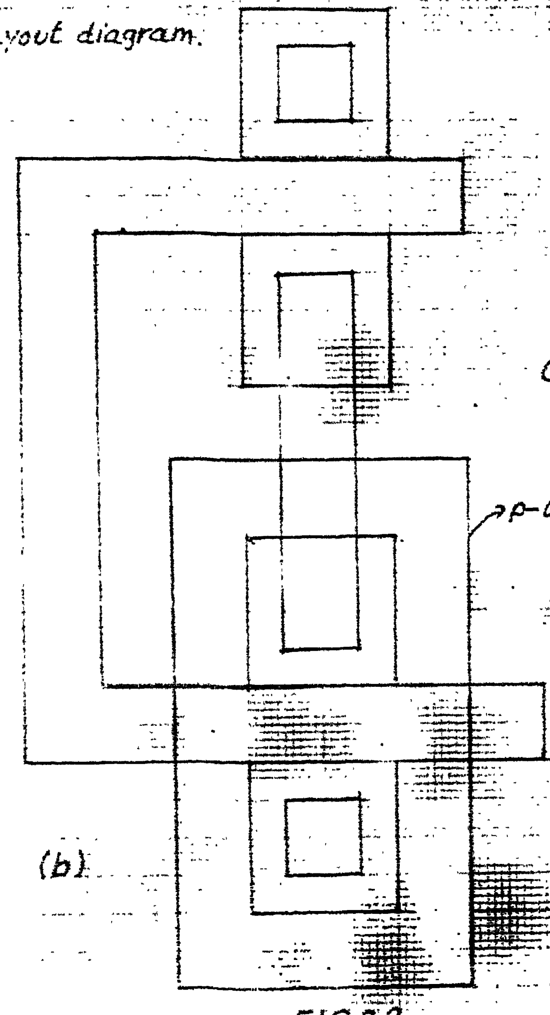


Circuit diagram of pseudo NMOS Inverter

FIG 2.3



Circuit diagram.
(a)



CMOS Inverter

(b)

FIG 2.2

```
+CJ=1.6E-4 CJSW=1.5E-10 UO=550 VTO=1.03 CGSO=1.33E-10
+CGDO=1.33E-10 NSUB=4E15 THETA=0.06 ETA=0.14 VMAX=12E4
+PB=0.7 MJ=0.5 MJSW=0.3 NFS=1E10
.MODEL PM PMOS LEVEL=3 RSH=100 TOX=250E-10 LD=0.1E-6
+XJ=0.6E-6 CJ=7E-4 CJSW=4.5E-10 UO=220 VTO=-0.66
+CGSO=5.5E-10 CGDO=5.5E-10 TPG=-1 NSUB=5E15 ETA=0.06
+THETA=0.03 VMAX=17E4 PB=0.7 MJ=0.5 MJSW=0.3 NFS=1E10
.END
```

The DC transfer characteristic is shown in Fig. 2.4. The transient response is shown in Fig. 2.5.

2.4 PSEUDO NMOS

This is a type of logic which uses PMOS as load device and NMOS as driver device. But here, as in CMOS, every NMOS device does not accompany a PMOS device. So the area requirement is less than CMOS. It is an effort to simulate CMOS characteristics without paying the price in area. Here the logic low (positive logic] is not zero volt, as the PMOS, with its gate grounded is not cutoff only saturated. Hence there will be a current through it. Fig. 2.3 shows the circuit diagram of such an inverter with the gate grounded. Its layout is similar to that of CMOS (only for an inverter) with only minor modifications. The SPICE input deck for such a device is shown below

```
NMOS PULLDOWN WITH PMOS LOAD WITH GATE GROUNDED

MPD 2 1 0 0 NM L=5U W=10U PD=40U PS=40U AS=100P AD=100P
MPU 2 0 3 3 PM L=10U W=5U PD=40U PS=40U AD=100P AS=100P
VSUP 3 0 5V
```

```
  C1 2 0 1PF
.DC VIN 0 5 0.25
.PRINT DC V(1) V(2)
.TRAN 10N 500NS
.PRINT TRAN V(1) V(2) I(VSUP)
.MODEL NM NMOS LEVEL=3 RSH=35 TOX=250E-10 LD=0.1E-6 XJ=0.14E-6
+CJ=1.6E-4 CJSW=1.5E-10 UO=550 VTO=1.03 CGSO=1.33E-10
+CGDO=1.33E-10 NSUB=4E15 THETA=0.06 ETA=0.14 VMAX=12E4
+PB=0.7 MJ=0.5 MJSW=0.3 NFS=1E10
.MODEL PM PMOS LEVEL=3 RSH=100 TOX=250E-10 LD=0.1E-6
+XJ=0.6E-6 CJ=7E-4 CJSW=4.5E-10 UO=220 VTO=-0.66
+CGSO=5.5E-10 CGDO=5.5E-10 TPG=-1 NSUB=5E15 ETA=0.06
+THETA=0.03 VMAX=17E4 PB=0.7 MJ=0.5 MJSW=0.3 NFS=1E10
.END
```

The DC transfer characteristic is shown in Fig. 2.4. The transient response is shown in Fig. 2.5.

2.5 THE BASIC GATES

2.5.1 For NMOS

The basic gates are NAND and NOR, they are shown in Fig. 2.6(a) and Fig. 2.6(b) for two inputs in the NOR gate when any of the inputs X and Y is high, there is a path for current to flow to the ground, resulting in low output. For both the case three transistors are required.

2.5.2 For CMOS

Here also the basic gates are NAND and NOR. They are show in Fig. 2.7(a) and 2.7(b) respectively. Here the number of gates increases to 4 for each gate. In either high or low con-

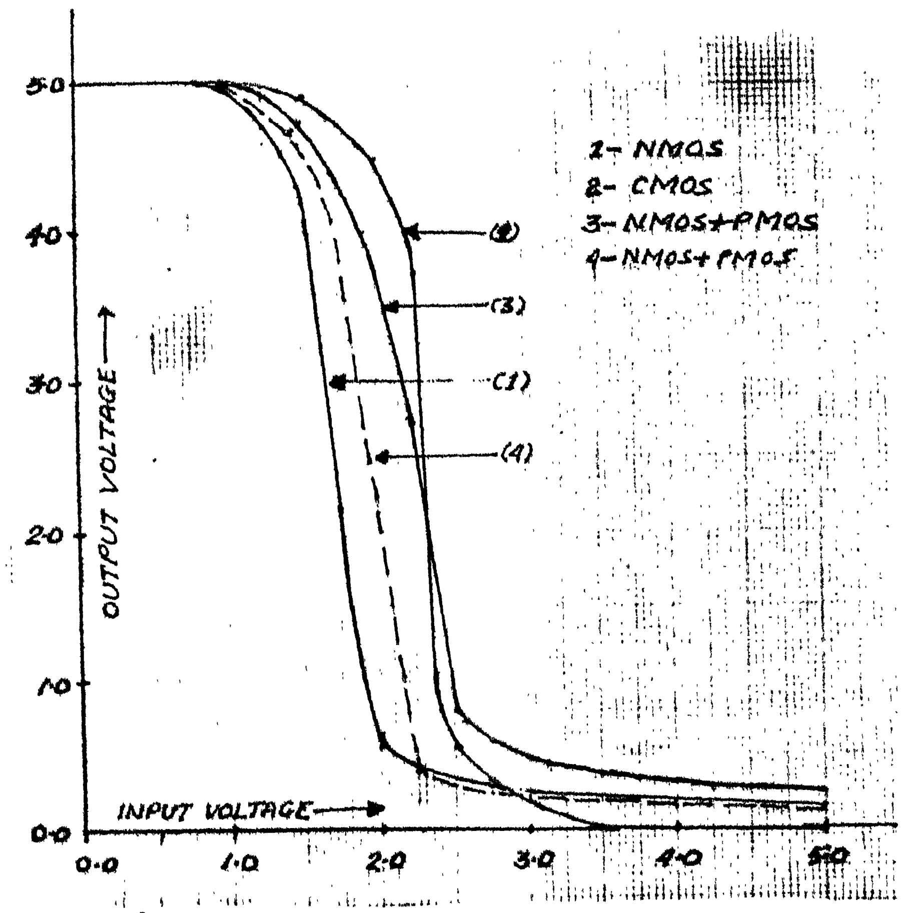


Fig. 2.4 DC Transfer Characteristic

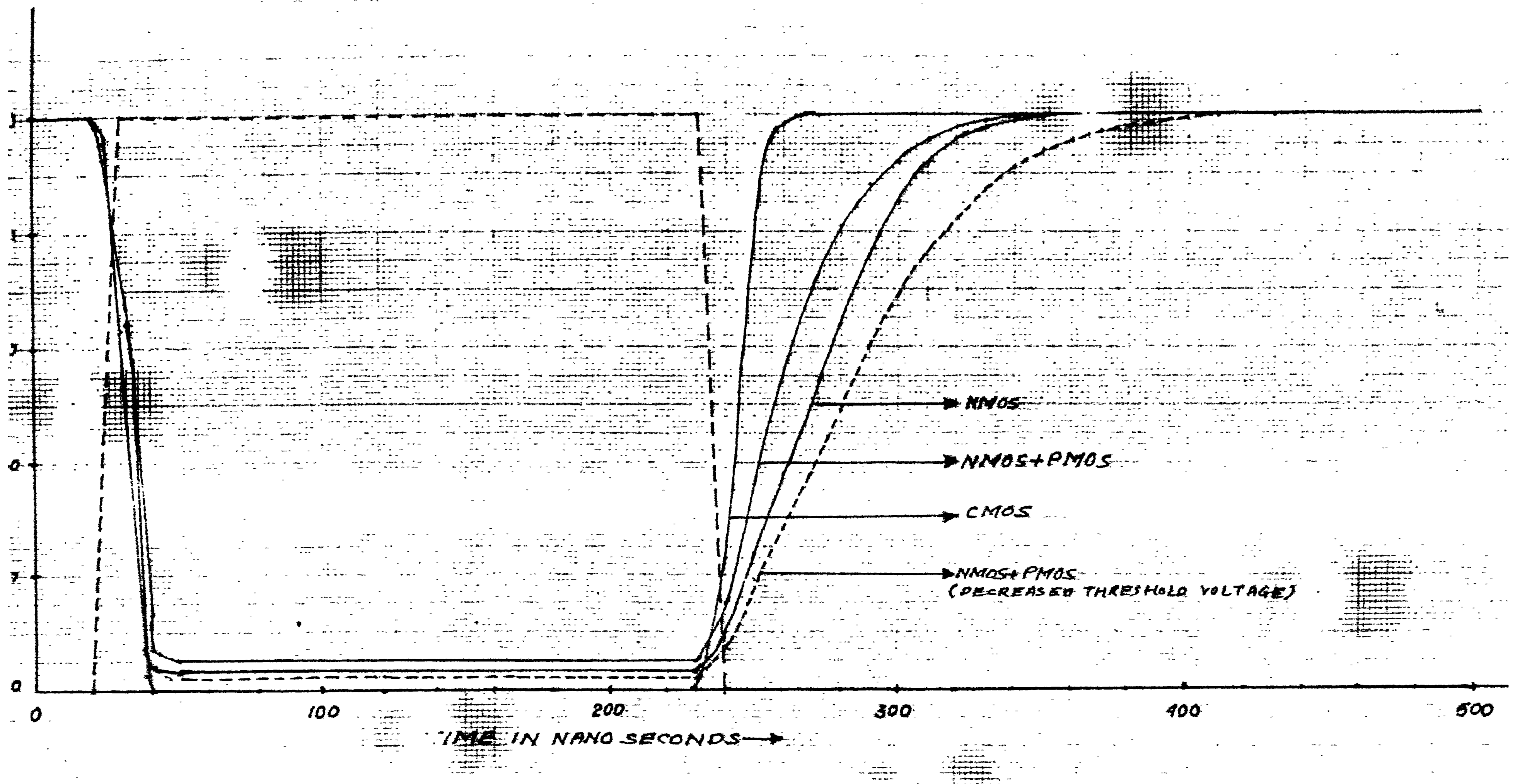


Fig. 2.5 Transient Response

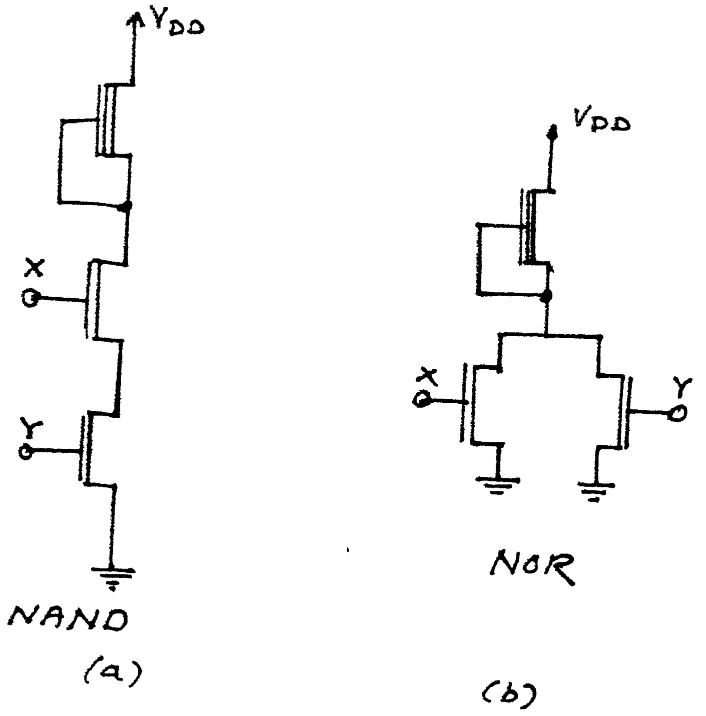


Fig.2.6 NMOS Gates

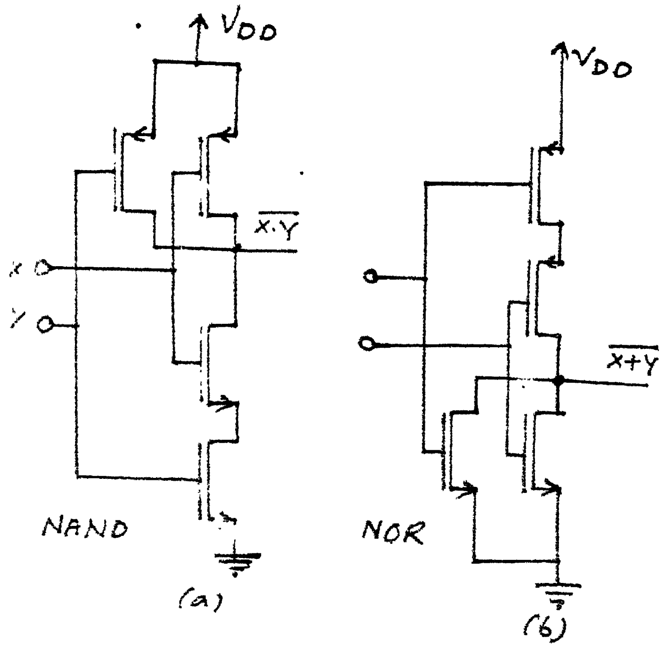
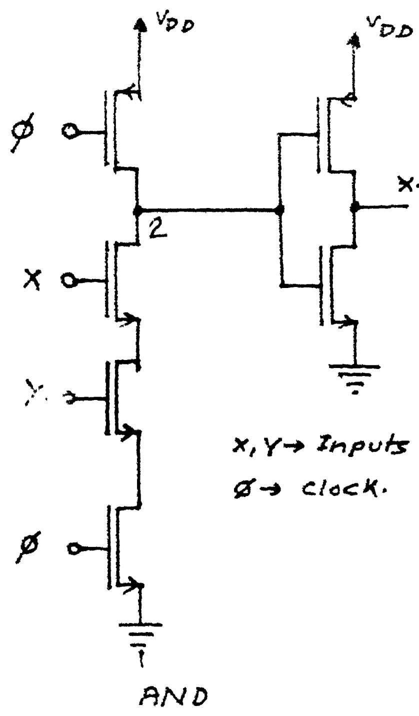


Fig.2.7 CMOS Gates

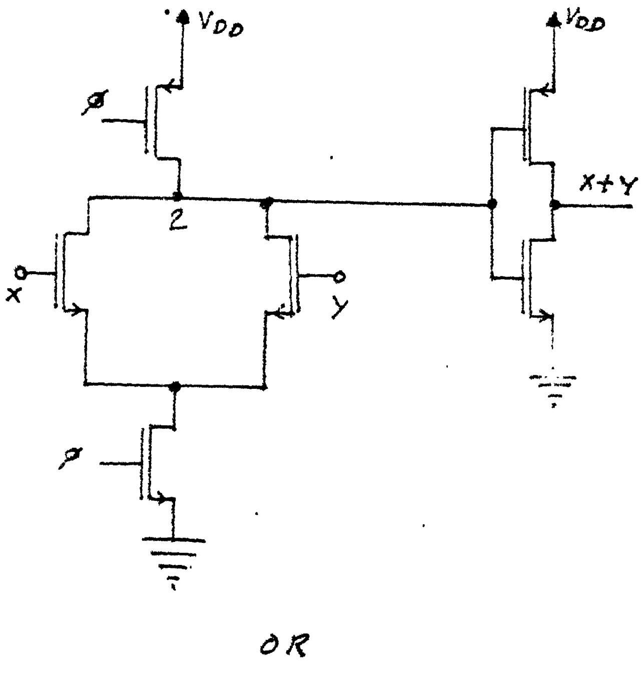
dition of output there is no current through the devices, curr-ent flows only in transition.

2.5.3 For Pseudo NMOS

This type of configuration is usually used in dynamic cir-cuits. One reason is the consumption of considerable power as shown by the SPICE analysis. The basic gates here are AND and OR [21]. Fig. 2.8(a) and 2.8(b) show the AND gate and the OR gate respectively, These type of circuits consist of two parts the dynamic part and the static CMOS inverter part. The CMOS inverter acts as a buffer. The source of the PMOS device in th dynamic part is connected to supply. The bottom NMOS has its gate connected to a clock. The gate of the PMOS is also conn-ected to the same clock. When the clock is low the bottom NMOS device is off, but the PMOS device is on, this results in prechanging of node 2 to V_{DD}. This period is called prechang-ing period. During this period the output of the CMOS buffer is low. When the clock Ø is high PMOS is off and NMOS is on. Then the node 2 dischanges or remains unchanged depending on the condition of the inputs. This period is called evaluation period. No type of inverted output (i.e. NOR, NAND or NOT) can be obtained from this type of gate. For this CMOS inverte is used. In practical circuits it is reported to be faster than CMOS [21]. One reason for this is that each output in CMOS has to drive twice the number of gates in pseudo NMOS. Hence the capacitance to be charged is reduced to half.



(a) Pseudo NMOS AND Gate



(b) Pseudo NMOS OR Gate

Fig. 2.8 Pseudo NMOS Gates

2.6 COMPARISON

Table 1 shows the characteristic of a typical inverter of each device . From this table it can be seen that for all the devices logic high is 5.0V. The logic low of the pseudo NMOS inverter is higher than the others. This can be a disadvantage in some situation. Its noise margin is slightly larger than that of total NMOS inverter. Progation delay of it is between that of CMOS inverter and NMOS inverter. This is a distinct advantage over NMOS inverter. From the area point of view NMOS is far superior to the others, especially to CMOS. From the power dissipation point of view CMOS is the best. In case of pseudo NMOS when the output is high the PMOS is on, when the output goes low the PMOS remains on too long. This can be modified by suitable threshold voltage for the PMOS. The gate here can not be connected to supply and also to the drain. The values after failuring of threshold voltage is shown in bracket. The dashed curve in Fig. 4 and 5 is for pseudo NMOS inverter with changed threshold voltage. From these it is seen that whereas dc characteristic has improved the transient response/has become very slow.

In real circuits pseudo NMOS devices are of the same speed as that of CMOS. Because in CMOS capacitance connected in to each output is double that of pseudo NMOS. Using dynamic gates the power consumption can be reduced very much. The saving in area is also significant . As each stage contains a buffer this gate is suitable for driving circuits with large fan out (such ALUs)

Table 1

Quantity	NMOS	Pseudo -NMOS	CMOS
1. Logic high, V_{oH} volts	5.0	5.0(5.0)	5.0
2. Logic low, V_{oL} volts	0.15	0.25(0.119)	0
3. Transition width volts	0.5	0.6(0.8)	0.5
4. Noise margin MN, volts	1.35	1.4(1.291)	2.0
5. Propagation Delay, nsec.	20	15.5(27)	9.5
6. Rise time, tr, nsec.	66	58(92)	16
7. Fabrication complexity	Simpler, 8 masks are needed	More complicated,10 masks are needed	More com licated masks ar needed.
8. Average Power dissipation, μW	225.8	402.5(136)	41.6
9. Area, ratio	1	2.4 (only for inverter)	2.4

Here PLAs with only NMOS devices are considered.

CHAPTER 3

DESIGN METHODOLOGY OF THE PLA

3.1 INTRODUCTION

As ICs are evolving ,their design methods are also evolving. Initially, when ICs contained about a hundred transitio their design was not a time consuming proposition. Till LSI customs design approach was found suitable, though a bit time consuming. To reduce design time designers took recourse to automation. In VLSI the number of devices is so large that partial automation of ICs becomes a necessity. For design automation structures having regularity are found to be suitable [1],[2],[3] and [5].

3.2 PLA DESIGN

Usually PLAs are parts of an LSI/VLSI system. They are used to realise combinational logic functions required in a chip. Hence connection of PLAs to other parts of the IC should be considered carefully in the over all design of the chip. PLAs are considered as macros. Macro is an aggregate of circuits which have high internal connectivity and appears together when physically implemented. They are used like NAND,NOR gates.

The workbench for designing PLA has a combinational logic minimizer, a logic simulator, a folder, a code generator for error detection, layout generator and a worst case path simulator [29]. PLA design consists of two parts, logical design and physical design. In logical design the function in the input specification

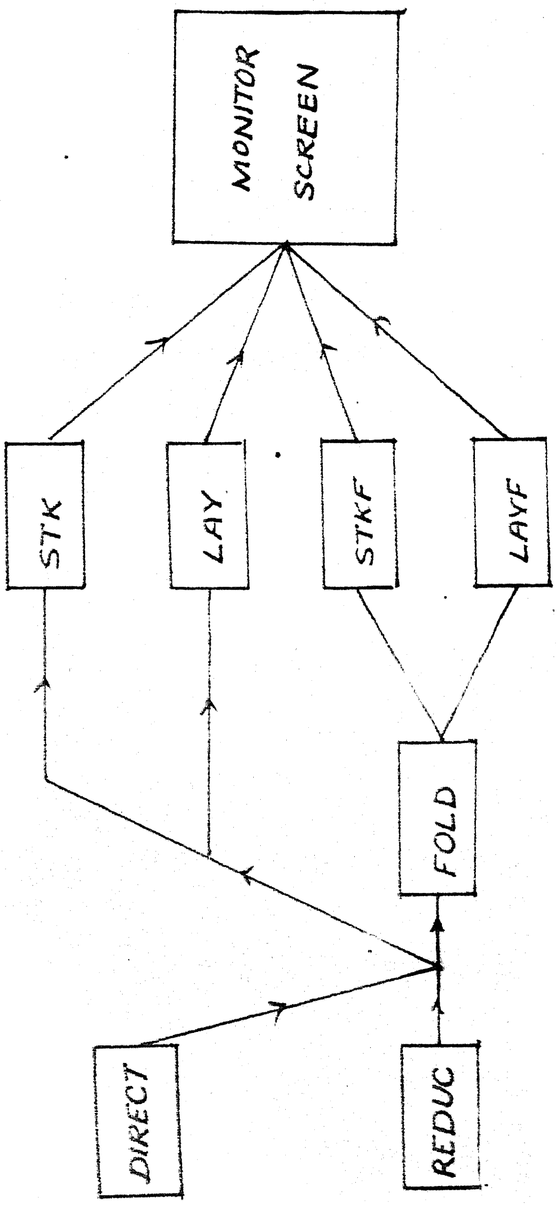


Fig. 3.1 PLAGE

is first minimized in the logic minimizer. Then the minimised expression for the function goes to the folder. After this the whole chip is simulated for logic verification in a logic simulator. If appropiate results are obtained in this stage then physical design is started. In physical design the layout of the PLA with other parts of the integrated circuit is generated. It is then checked if all the design rules are followed or not. At this stage the circuit characteristics of the IC is evaluated. The worst case path simulator mentioned above gives the maximum delay and other circuit properties of the PLA.

3.3 PLAGE

Here a subset of such a system is described. It contains a logic minimizer, a folder (only column), stick diagram generator and layout diagram generator. The name of the system is PLAGE (PLA Generator]. Fig. 3.1 shows the relation between the PLA design aid programs in PLAGE. They are seven in number.

The program REDUC is for combinational logic minimization. The criterion taken here for minimising logic function is the number of product terms. This criterion is very suitable for PLAs. The output from this program can be input to the programs in the next stage as shown in Fig. 3.1.

FOLD is a program for folding a PLA. Its input can come either from DIRECT or REDUC. The program outputs the column pairs that can be folded along with a compatible row ordering.

The program STK can draw the stick diagram of an unfolded PLA. Its input is the output of REDUC or DIRECT. STKF can draw the stick diagram of a folded PLA. Its input is the output of program FOLD. The output of both of these program is a graphic monitor.

LAY can draw the layout diagram of an unfolded PLA. Similarly LAYF can draw the layout diagram of a folded PLA.

All of these programs have been implemented in the ND-500 Computer of the CAD centre of I.I.T., Kanpur. All the programs are written in PASCAL. As no graphic package in PASCAL is available, these are written with the help terminal graphic facilities. The terminals available are Tektronic 4100 series graphic terminals.

The REDUC program has been used successfully for 6 input 6 out-
t combinational logic functions. FOLD has been used success-
ly for 50 columns PLAs. It has been seen that the display
terminals is not very good for PLAs larger than 20 columns.

The criterion for logic minimization can be changed so that
e number of folding obtained is larger thus minimising the area most. This can be done by adding the second criterion that the number of X_s in the personality matrix is a maximum. This will increase the probability of folding.

The input and output formats of the programs are explained in the Appendix A.

CHAPTER 4

MINIMIZATION

4.1 THE NECESSITY OF LOGIC MINIMIZATION

Logic minimization is an important part of PLA design, for otherwise considerable area will be wasted if the product terms are minterms. Logic minimization can reduce the number of rows in a PLA drastically. Without minimization folding also becomes less effective.

4.2 METHODS USED FOR MINIMIZATION

Broadly the methods can be divided into two types. One is heuristic and the other is deterministic. Deterministic methods are always able to find out the optimal minimization,though in doing so it may take large amount of time. Among these are those in [7], [8], [11], [22] and [23]. The other type is heuristic. They are able to find out a near optimal solution but in less time. Heuristic methods are computationally more complex. Examples of heuristic methods are MINI [24], PRESTO [25], PRONTO [26] and ESPRESSO II [27].

4.3 DEFINITION OF TERMS [9]

I :- stands for non inverted variable

O :- stands for inverted variable

X :- don't care

Prime Implicant :- A prime implicant P of a combinational logic function f is a product term which is covered by f, and deletion of a literal from which results in a new product term which is not covered by f.

Cover :- A set of prime implicants which contains all the minterms of a function.

Substraction :- Let p and q be two product terms. Then p-q means the set of minterms which are contained in p but not in q. Thus if for a single output function we have the product terms IXOX and IOOO, then IXOX-1000 = 110X, 1XOI , here the set of minterms is expressed in terms of prime implicants.

Contain :- A product term contains another if all the minterms of the second is also contained in the first. If a contains b then b$\varnothing$a.

4.4 DESCRIPTION OF THE MINIMIZATION METHOD

4.4.1 For Single Output Function

The method will be described with an example. The Karnaugh map (K-map) of Fig. 4.1 defines the function.

The method is divided into two parts . The first part finds out the prime implicants and the second part finds out the minimal cover.

	00	01	11	10
00	0_0	0_1	I_3	0_2
01	0_4	I_5	I_7	I_6
11	I_{12}	I_{13}	I_{15}	a_{14}
10	I_8	I_9	I_{11}	I_{10}

Fig. 4.1

The criterion selected for minimization is the number of product of terms, [10], [22] and [28]. The starting point of the first part of the procedure is the choice of the initial product term as XXXX. Now from this product term one maxterm is substracted. (i.e. those in the K-map with O). In the example the first subtrahend is OOOO. After substraction the minuend is deleted from the list of product. We get the list in L1. Now from the K-map another maxterm is selected and deleted from this list. The choosen maxterm is OOOI.

Now in list L1, term 1 does not contain this maxterm, hence remains unaltered. So is for terms2 and 3. Term 4 contains the choosen maxterms. Hence it is substracted from term 4. Term 4 is deleted from the list and the terms resulting from substraction is added to the list. This is shown in L2. Starting from L2, the list should be cleared of redundant terms. In L2 we have term 4¢ term 1, term 5¢ term 2 and term 6¢ term 3. So terms 4,5 and 6 can be deleted from the list. The list we get after this is shown in L2'. The next subtrahend choosen is OOIO. We have OOIO ¢ term 3. After substraction we get the list in L3. In this list we have term 4¢ term 2 and term 5¢ term 1. So terms 4 and 5 can be deleted from the list. The modified list is L3'. The next term to be subs-tracted is OIOO. It is contained only in term 2 of L3'. After substraction we get the list L4. In L4 we have term 3¢ term 1. After deletion of term 3 we get the

```
1IXXX
2XIXX
3XXIX
4XXXI
 L1

1IXXX
2XIXX
3XXIX
4IXXI
5XIXI
6XXII
 L2

1IXXX
2XIXX
3XXIX
 L2'

1IXXX
2XIXX
3XXII
4XIIX
5IXIX
 L3

1IXXX
2XIXX
3XXII
 L3'
```

```
1IXXX
2XXII
3IIXX
4XIIX
5XIXI
 L4

1IXXX
2XXII
3XIIX
4XIXI
 L4'
```

the list L4'. The last subtrahend is 1110. This is contained in terms 1,2 and 3 of L4'. The resubting list is L5. In L5 term 6⊄ term 4, term 7. Hence term 6 is deleted. The resulting list is shown in L5. The terms in L5' are the prime implicants.

```
1IOXX
2IXOX
3IXXI
4XXII      IOXX
5OIIX      IXOX
6XIII      IXXI
7XIXI      XXII
 L5        XIXI
           OIIX
            L5'
```

These are shown in the K-map of Fig. 4.2

	00	01	11	10
00	O_0	O_1	I_3	O_2
01	O_4	I_5	I_7	I_6
11	I_{12}	I_{13}	I_{15}	I_{14}
10	I_8	I_9	I_{11}	I_{10}

Fig. 4.2

The second part finds out the minimal cover with respect to number of product terms. To each term of the prime implicant list a weight is attached. This is the number of minterms not yet covered but contained in the prime implicant. Let B be a set containing the minterms, not yet covered. A be a set containing all the minterms covered till now. Initially B contains all the minterms and A is empty. If C_i is the set of minterms contained in the prime implicant and W_i is the weight at present of the prime implicant then.

$W_i = |C_i \cap B|$ (where $|x|$ implies the number of elements in X).

The initial situation is shown in Fig. 4.3.

W	Prime Implicants	
4	10XX	B = 3,5,7,6,12,13,15,8,9,10,11
4	1X0X	A = ∅
4	1XXI	
4	XXII	
4	XIXI	
2	011X	

Fig 4.3

Now from this list one prime implicant with the highest weight is selected. This added is to a list called selected. Then the set A and B are adjusted. B is modified to contain all those minterms not contained in any selected prime implicants. (i.e. $B \leftarrow B - B \cap C_i$) and A is modified to contain all the minterms that are covered by the prime implicants choosen till now. (i.e. $A \leftarrow A \cup C_i$). This is shown in Fig. 4.4.

W	Prime Implicant		selected
0	10XX	B = 3,5,7,6,12,13,15	10XX.
2	1X0X	A = 8,5,10,11	
2	1XXI		
3	XXII		
4	XIXI		
2	011X		

Fig. 4.4.

The situation after each selection is shown in Fig. 4.5. This process of selecting the prime implicant with highest weight and readjusting the sets, weights and lists is continued till A = ∅. (i.e. when all the weights are 0).

W	Prime Implicants
0	10XX
1	1X0X
0	1XXI
2	XXII
1	011X
0	XIXI

B = 3,6,12
A = 8,9,10,11,5,7,13,15

selected
10XX
X1XI

W	Prime Implicants
0	10XX
0	1X0X
0	1XXI
1	XXII
0	XIXI
1	011X

B = 3,6
A = 8,9,10,11,5,7,13,15

selected
10XX
XIXI
1X0X

W	Prime Implicants
0	10XX
0	1X0X
0	1XXI
0	XX11
0	XIXI
I	011X

B = 6

A = 8,9,10,11,9,7,13,15,3

selected
10XX
XIXI
1X0X
XXII

W	Prime Implicants
0	10XX
0	1X01
0	1XX1
0	XXII
0	XIXI
0	0IIX

B = ∅

A = 8,9,10,11,5,7,13,15,3,6

selected
10XX
XIXI
1X0X
011X
XXII

Fig. 4.5

4.4.2 <u>For Multiple Output Function</u>

For multiple output function also the method is similar but be here another part called Tag is attached to each prime implicant and each subtrahend (maxterms) to indicate their presence or absence in an output. In case of subtrahend an I means the subtrahend is a maxterm of that particular output an X means the subtrahend is a minterms. In case of product terms an I implies the presence the product term in the output and an X implies absence. For a output there can be four combinations of I and X of tags of product term and subtrahend. They are I and I, I and X, X and I and X and X,. The first stands for product term. I and I implies the subtrahend is a maxterm and the product term is contained in the output. If the input part of subtrahend is contained is the input part of the product term then there resultants input parts are same as that of single output function but the output tag having X where there is a (I and X), (X and X) and (X and I) combination. There will be one additional resultant with the tag having I where there were (I and X) combinations; and all others X; the input part remains unaltered. The procedure is explained with an example. The function is defined in the K-maps of Fig. 4.6.

	00	01	11	10
00	0_0	1_1	0_3	1_2
01	1_4	1_5	1_7	0_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	0_8	1_9	1_{11}	1_{10}

	00	01	11	10
00	1_0	1_1	0_3	1_2
01	0_4	1_5	1_7	1_6
11	1_{12}	1_{13}	1_{15}	0_{14}
10	0_8	0_9	0_{11}	1_{10}

	00	01	11	10
	1_0	1_1	1_3	0_2
	1_4	1_5	1_7	0_6
	0_{12}	1_{13}	1_{15}	0_{14}
	1_8	0_9	0_{11}	1_{10}

The maxterms with tags are

```
0000IXX
0010XXI
001111X
0100XIX
01101XI
1100XXI
1110111
100011X
1001XII
1011XII
```

Fig. 4.6

Substracting 0000IXX from XXXXIII we get

```
XXXX'XII
1XXX'1XX
X1XX'1XX
XXIX'1XX
XXXI'IXX
  L1
```

Next substracting 0010XXI from L1 we get list L2.

```
1XXXX'XIX
21XXX'1XX
3X1XX'1XX
4XXIX'1XX
5XXXI'1XX
61XXX'XXI
7XIXX'XXI
8XXOX'XXI
9XXXI'XXI
   L2
```

```
1XXXX'XIX
21XXX'1XI
3XIXX'1XI
4XXXI'1XI
5XXIX'1XX
6XXOX'XXI
   L2'
```

In L2 2 and 6 are similar in their input parts only hence they can be replaced by a single term containing the same input part but different output part. The tag part will have an I whenever either of the tags of 2 or 6 has an I. Similar is the case for 3 and 7 and 5 and 9. After doing these we get the modified list L2'. The operation just described can be represented by :=. So we have 2:=6 ; 3:=7 ; 5:=9;

Next 0011 LLX is substracted. This maxterm is contained in terms 1,4 and 5 of L2'. Substraction from term 1 yields the terms 4,5,6 and 7 in L3. Substracting from 4 in L2 we get the terms 8,9,10 and 11 in L3. 11 is the term where we get the I

```
 11XXX'1XI
 2XIXX'1XI
 3XXOX'XXI
 41XXX'XIX
 5XIXX'XIX
 6XXOX'XIX
 7XXXO'XIX
 81XXI'1XX
 9XIXI'1XX
10XX01'1XX
 11XXXI'XXI
121XIX'1XX
 13XIIX'1XX
14XX10'1XX
```

L3

```
11XXX'111
2XIXX'111
3XXOX'X11
4XXXO'XIX
5XX01'1XX
6XXXI'XXI
7XX10'1XX
```

L3'

```
 11XXX'111
 2XX01'1XX
 3XXXI'XXI
 4XXIO'1XX
 5XIXX'1XI
 611XX'XIX
 7X11X'XIX
 8XIXI'XIX
 9XXOX'XXI
101XOX'XIX
11XOOX'XIX
12XX01'XIX
131XXO'XIX
14XOXO'XIX
15XX10'XIX
```

L4

and X combination mentioned above. Substracting from 5 we get the terms 12, 13 and 14 in L3. Now one terms is contained in another if it has I and O in all the places in input (reverse for output) where the second term has them. If a is contained in b then it is written a⊄b. Also in some cases the input parts may satisfy the just mentioned condition but tag part may not. In that case if there is an I and I combination in tag the corresponding I in a is converted to an X. This can be represented by a $\overset{*}{*}$. For L3 we have 13⊄2, 8⊄1, 9⊄2, 12⊄1, 1:=4, 2:=5, 3:=6. After doing these we get L3'. Next O1OOXIX is substracted. We get list L4. Substracting from 2 we get 5, 6, 7 and 8 of list L4. From 3 we get 9, 10, 11 and 12 of L4. From 4 we 13, 14 and 15. In L4 we have 6⊄1, 10⊄1, 13⊄1, 2:=12 and 4:=15. Then we get list L4'.

```
 11XXX'111
 2XX01'11X
 3XXXI'XX1
 4XX10'11X
 5XIXX'1XI
 6XIIX'XIX
 7XIXI'XIX
 8XXOX'XXI
 9XOOX'XIX
10XOXO'XIX
```

L4'

Substracting 01101XI from L4' we get L5 and L5'.

```
 1 1XXX'111                      1 1XXX'111
 2 XX01'11X                      2 XX01'11X
 3 XXX1'XX1   9C1                3 XXXI'XXI
 4 X1X1'XIX  11C1                4 XIXI'111
 5 XX0X'XXI  4:=14               5 XX0X'XX1
 6 X00X'XIX                      6 X00X'XIX
 7 X0X0'XIX                      7 X0X0'XIX
 8 XX10'XIX                      8 XX10'XIX
 9 1X10'1XX                      9 X010'1XX
10 X010'1XX                     10 X10X'1X1
11 11XX'1X1                     11 X11X'XIX
12 X1XI'1X1
14 X11X'XIX                          L5'

     L5
```

Substracting 100,XXI from L5' we get L6 and L6'.

```
 1 XX01'11X        12C2          1 XX01'11X
 2 XXXI'XXI         2*3          2 XXXI'XXI
 3 XIXI'111        15C2          3 XIXI'11X
 4 X00X'XIX        17C2          4 X00X'XII
 5 X0X0'XIX        16C18         5 X0X0'XIX
 6 XX10'XIX        4:=14         6 XX10'XIX
 7 X010'1XX                      7 X010'1XX
 8 X11X'XIX                      8 X11X'XIX
 9 1XXX'11X                      9 1XXX'11X
10 10XX'XXI                     10 10XX'XX1
11 1XIX'XXI                     11 1XIX'XX1
12 1XXI'XXI                     12 0X0X'XX1
13 0X0X'XXI                     13 X10X'1XX
14 X00X'XXI
15 XX01'XX1                          L6'
16 010X'XX1
17 X111'XX1
18 X10X'1XX

     L6
```

Continuing in this manner ultimately we will get

```
                 1 XIXI'III          12 10X0'XX1
                 2 X010'11X          13 0X01'XIX
                 3 0X0X'XXI          14 L01X'1XX
                 4 X10X'XIX          15 1XX1'1XX
                 5 0X10'XIX          16 X000 'XXI
                 6 011X'XIX
                 7 000X'XIX
                 8 00X0'XIX
                 9 110X'11X
```

The second part of the procedure finds out the minimal cover with respect to number of product terms. This part also is similar to the corresponding part of a single output function. Here the number of set A and B is n_o, no. of output, each. Let W_i be the weight of prime implicant i and let j be any output. Also let e_{ij} be a number which is 1 if prime implicant i belongs to output o_j. The following expression evaluates the weight.

$$W_i = \sum_{j=1}^{n_o} e_{ij} \; |B_j \cap C_i|$$

where $|B|$ gives the number of elements belonging to set B.

Set A_j contains all the minterms belonging to output j, which have been included in some prime implicant i (belonging to j) selected so far for the minimal cover. The union of A_j and B_j gives all the minterms of the output j.

After each selection set A_j and B_j is modified. Let A_j' and
the
B_j' be/modified sets then

$$A_j' = A_j \overset{\cap}{\#} C_i \text{ , for all } j=1 \text{ to } n_o \text{ and } e_{ij} = 1.$$

$$B_j' = B_j - C_i \text{ , for all } j=1 \text{ to } n_o \text{ and } e_{ij} = 1.$$

The Initially we have

$$B_j = C_i \text{, for all prime implicant having } e_{ij} = 1.$$

$$A_j = \emptyset \; .$$

After the weights have been calculated and set A_j and B_j initialized, the prime implicant with the highest weight is selec-

The next prime implicant to be selected is X01011X. Continuing in this manner the final situation in Fig. 4.9 will be obtained.

W		
0	XIXI	111
0	X010	11X
0	0X0X	XXI
0	X10X	1XX
0	0X10	X1X
0	011X	X1X
0	000X	XIX
0	00X0	XIX
0	110X	11X
0	XX01	1XX
0	0XX1	XX1
0	X000	XX1
0	10X0	XX1
0	0X01	X1X
0	101X	1XX
0	1XX1	1XX

$B_1 = \emptyset$

$A_1 = \{1,2,4,5,7,12,13,15,9,11,10\}$

$B_2 = \emptyset$

$A_2 = \{0,1,2,5,7,6,12,13,15,10\}$

$B_3 = \emptyset$

$A_3 = \{0,1,3,4,5,7,13,15,8,10\}$

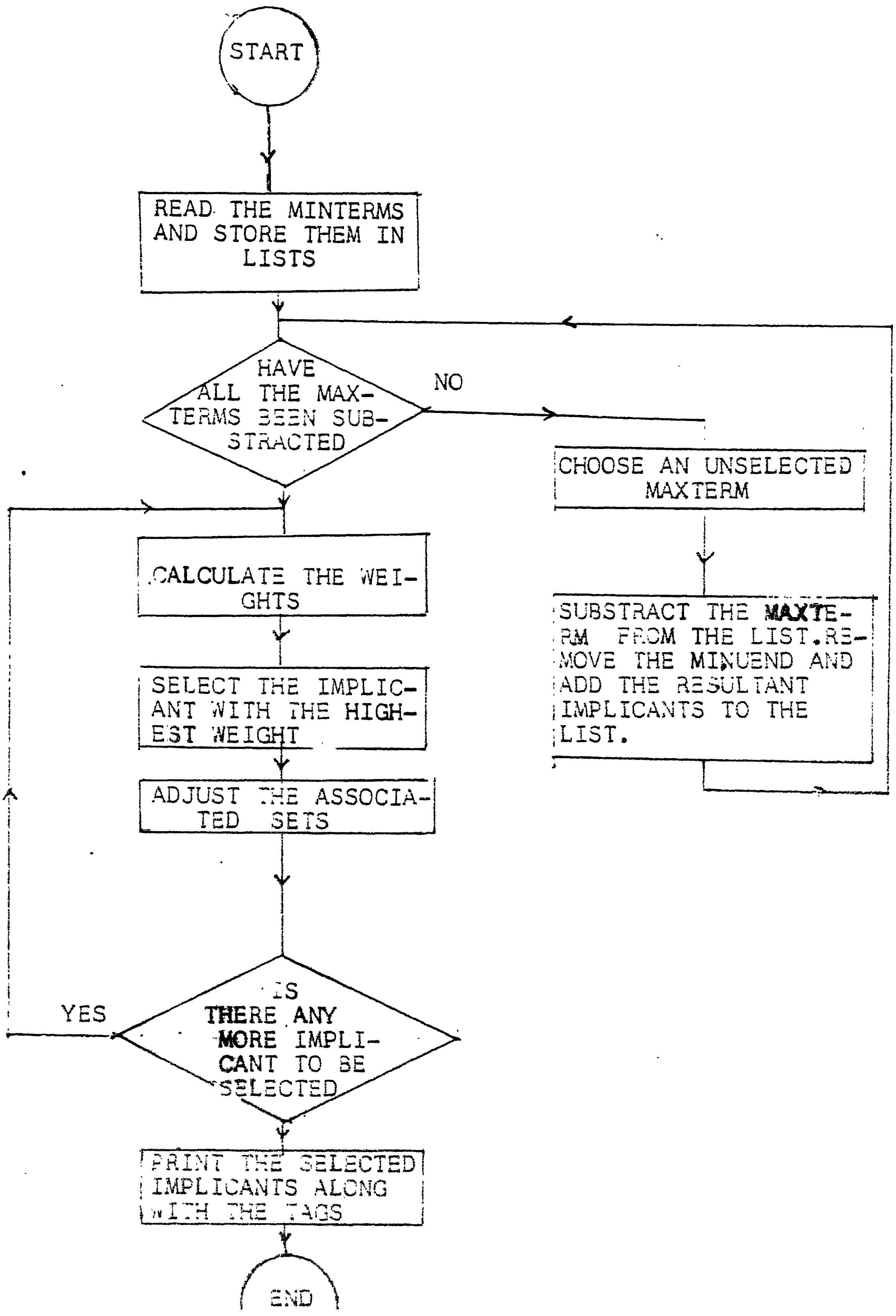
Selected

XIXIIII
X01011X
0X0XXXI
X10XIXX
000XXIX
XX011XX
10X0XX1
110X11X
0XXIXXI
101X1XX

Fig. 4.9.

This selection procedure does not always give the minimum number of terms. This type of procedure is called Greedy Procedure. In some problems such a greedy strategy pays off. In these, the best local solution is also the best global solution. But is not neccessarily true for every problem. There is a similar problem in Set Theary called Set Cover [17], where this greedy method does not always work . These things make the above procedure heuristic.

FLOW DIAGRAM OF REDUC


START
READ THE MINTERMS AND STORE THEM IN LISTS
HAVE ALL THE MAX-TERMS BEEN SUB-STRACTED
NO
CHOOSE AN UNSELECTED MAXTERM
SUBSTRACT THE MAXTERM FROM THE LIST. REMOVE THE MINUEND AND ADD THE RESULTANT IMPLICANTS TO THE LIST.
CALCULATE THE WEI-GHTS
SELECT THE IMPLIC-ANT WITH THE HIGH-EST WEIGHT
ADJUST THE ASSOCIA-TED SETS
IS THERE ANY MORE IMPLI-CANT TO BE SELECTED
YES
PRINT THE SELECTED IMPLICANTS ALONG WITH THE TAGS
END

CHAPTER 5

FOLDING

5.1 INTRODUCTION

This chapter explains the folding procedure. Folding is done to reduce the silicon area occupied by a PLA. Usually the personality matrix of a PLA contains a great deal of dont care terms (X). This means that a large amount of area is wasted only for interconnections. This wastage is not desirable, hence these Xs should not be implemented in the PLA. To do this folding is done. If the personality matrix of a PLA consists of about 40% Xs, then only 60% of the PLA is used for active devices. Folding splits a physical line into two or more segments so that different signals may share a common physical vertical or horizontal track. The objective of folding is to obtain on arrangement of vertical and horizontal lines so that the combinational logic function can be implemented into the smallest possible physical array [12]. The problem is shown to be NP complete [13]. So it is almost impossible to find out an efficient solution. Fig. 5.1 gives an example of folding.

If a PLA contains n_c columns and n_r rows then the area occupied by the PLA is proportional to $n_c x n_r$. Now if n_{rf} pairs of rows and n_{cf} pairs of columns are folded then the area of the resulting PLA is $(n_c-n_{cf})x(n_r-n_{rf})$. Here only column folding is considered.

5.2 DEFINITION OF TERMS

The ith row is indicated by r_i and the j column is indicated by c_j.

Two rows r_i and r_j are said to be disjoint if they do not have I or O in the same column. Thus in Fig. 5.1 r_3 and r_2 are disjoint. Two columns c_i and c_j are said to be disjoint if they do not have I or O in the same row.

A row r_i is said to be adjacent to a column c_j if the $r_i c_j$ the element of the personality matrix is I or O.

Thus r_i is adjacent to c_j

if $Pm\ [r_i, c_j]$ = I or O. (Pm is the personality matrix)

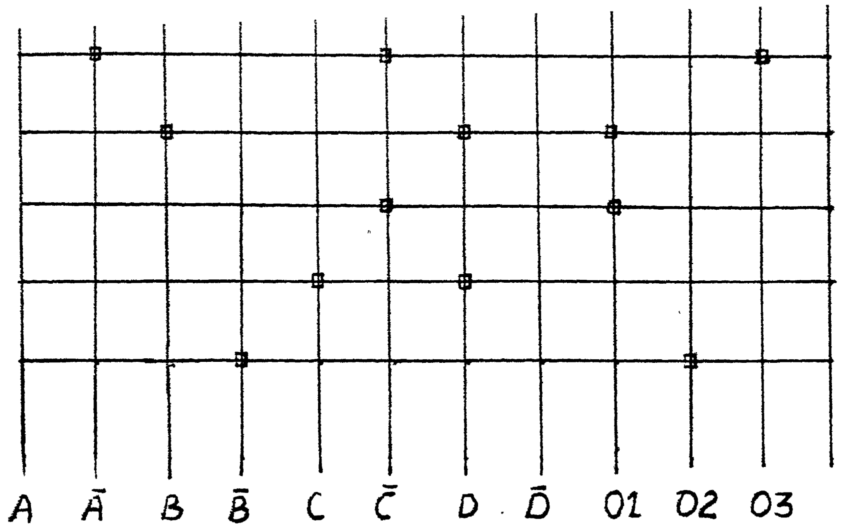
Ri denotes the set of rows adjacent to column c_i i.e.

$R_i = r_j | r_j$ is adjacent to c_i. C_i denotes the set of columns adjacent to row r_i, $C_i = c_j | c_j$ is adjacent to r_i .

Thus two columns c_i and c_j are disjoint if $R_i \quad R_j = \emptyset$.

An ordered folding pair $[c_i, c_j]$ indicates that c_i and c_j are foldable and c_i is placed above c_j. The folding of a pair of columns $[c_i, c_j]$ introduces the constraint that all the rows adjacent to c_i should be placed over all the rows adjacent to c_j. This is indicated by $R_i > R_j$, also if $r_{ki} \in R_i$ and $r_{kj} \in R_j$ then $r_{ki} > r_{kj}$.

A column disjoint graph is a graph associated with the columns of a PLA. Its vertices (V) is the set of columns and its edges (E) is the set of disjoint columns. It is indicated by G(V,E). $E = \{(c_i, c_j) | c_i$ and c_j are disjoint$\}$. To represent the ordered folding pairs a mixed (with directed and undirected

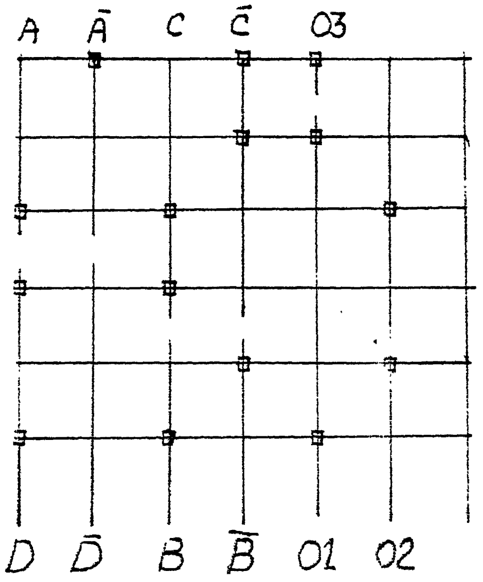


UNFOLDED PLA

(a)

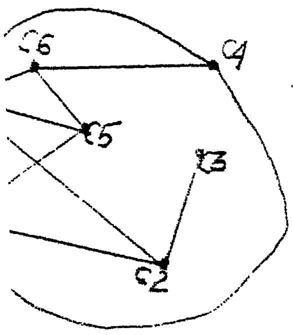
	C_1	C_2	C_3	C_4	C_5	C_6	
r_1	I	X	I	X	X	I	X
r_2	O	X	O	X	X	X	I
r_3	X	I	X	I	I	X	X
r_4	X	X	O	X	I	X	X
r_5	X	X	I	I	X	X	X
r_6	X	O	X	X	X	I	X

Personality Matrix



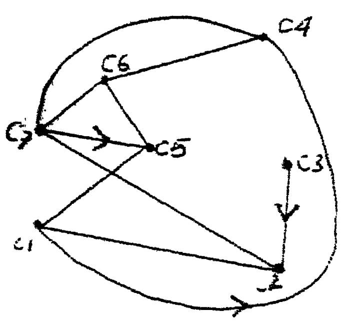
FOLDED PLA

(b)



UMN DISJOINT GRAPH

(c)



(d)

Fig. 5.1

edges) type of graph is constructed. Its set of vertices is V. The set of ordered folding pair (A) constitutes its directed edges. The remaining of the edges representing disjoint columns forms the set of undirected edges (E*). It is represented by G*(V,E,A*). Column disjoint graph of the PLA in Fig.5.1(a) is shown in Fig.5.1(c)also is shown one of its mixed disjoint graphs. in Fig. 5.1(d).

5.3 METHODS USED FOR FOLDING

Methods usually used for folding can be broadly divided into two types. One type uses Branch and Bound type of deterministic approach [14,15]. The other type uses Graph Theoretic algorithms. Deterministic algorithms can find out the optimal solution if sufficient time is allowed, which can be prohibitively large. Algorithms based on Graph Theory, on the other hand, can find out a near optimal solution in relatively less time. Hence use of this type of algorithms seems to be preferable.

5.4 GRAPH THEORETIC BASIS OF THE FOLDING PROBLEM

The objective here is to find out as many compatible ordered folding pairs as possible [16]. Alternating path (AP) :- An AP is a sequence of vertices, K = (V_1, V_2, V_3, V_m) of the mixed disjoint graph G*(V,E*,A) such that (V_{2p-1}, V_{2p}) A, for p=1,2,..m and (V_{2p}, V_{2p+1}) E* for p=1,...(m-1).

Theorem 1 : If two folding pairs are compatible then there must

must be a alternating path containing the directed edges representing the pairs in the mixed disjoint matrix.

<u>Proof</u> :- Let $[V_1,V_2]$ and $[V_3,V_4]$ be the directed edges. (i.e $[V_1,V_2]$, $[V_3,V_4]$ A).

They introduces the constraint $R_1>R_2$ and $R_3>R_4$. Now let us assume that they are foldable but there is no alternating path connecting them. This implies that (V_1,V_4) and (V_2,V_3) E^*. Which in turn implies that R_1 $R_4 \neq \emptyset$ and R_2 $R_3 \neq \emptyset$. So there exists some row r_a and r_b such that r_a R_1 R_4 and r_b R_2 R_3. From which we get $R_3>r_a>R_2$. So we have

$$r_b>r_a>r_b.$$

The above expression says that row r_a is above r_b and /at the same ti also below r_b. Which certainly is not possible. Hence a necessary condition for compatibility of folding of two pair is that there must be an alternating path.

Another thing which naturally follows is that for compatibility of more than two folding pairs there must be an alternating path through each pair of foldable pairs.

<u>Theorem 2</u> :- A necessary/condition for compatibility of a set of foldable pairs is there must be an alternating path containing all the directed edges.

<u>Proof</u> :- This/proved by induction on the length of the alternating path. One condition for implementability of an AP is each pair of foldable pair must have an alternating path through them.

Let there be an alternating path with K directed edges. Let there be another directed edge compatible with the other directed edges. The theorem is proved by showing that there exists an alternating path of length K+1 containing the directed edge. Let the existing alternating path be V_1 U_1 - V_2 U_2 V_3 U ----- V_q U_q . V_{K+1} U_{K+1} is the (K+1)th directed edge. As it is compatible with other directed edges V_i U_i i=1,---K, there should be an undirected edge, either (U_i-V_{K+1}) or $(U_{K+1}-V_i)$ for each i, by Theorem 1. In case either (U_{K+1},V_1) or (U_K,V_{K+1}) edge exists then there exists an alternating path (Fig. 5.2(a)). If neither of these edges exists then (U_1-V_{K+1}) and (U_{K+1},V_K) exists. With these edges an edge (U_{K+1},V_2) completes an alternating paths

V_1 U_1 - V_{K+1} U_{K+1} - V_2 U_2 --- V_K U_K . (Fig. 5.2(b)).

Now if this edge also does not exist, then an edge (V_3-U_{K+1}) completes an alternating path. If this does not exist, and continuing in this manner till we reach U_{K+1}. But here we get the alternating path V_1 U_1-V_2 U_2 --- V_{K-1}- U_{K-1}-V_{K+1} U_{K+1}-V_K U_K as shown in Fig. 5.2(c).

The base case here is an alternating path with two directed edge, from Theorem 1. Hence there must exist an alternating path. It is a necessary condition, its satisfaction is not sufficient for compatibility of folding.

A folding set is said to induce cyclic condition if there exists two rows r_a and r_b such treat

$$r_a > r_b > r_a.$$

Theorem 3:- The optimal column folding problem is equivalent to finding a longest AP on the associated mixed disjoint graph, such that the path does not induce cyclic ordering on rows.

Proof :- As each directed edge in if represents an ordered folding pair and the longer path contains more directed edges, the longest AP will represent the optimal column folding.

5.5 PROCEDURE FOR FINDING PATH

This is the most critical part of the procedure. Each point of the path is choosen probabilistically. First a path with no directed edge if found out. Let 1 —— K be such a path (Fig. 5.3(a)). K is called the current end vertex. The path is incremented as follows.1)The next vertex (K+1) is choosen probabilistically from the set of the vertices to which K is connected by an undirected edge [16]. 2) Now depending on the status of K+1. The following things are done. a) If K+1 is not on the path then it is added to the path. b) If it is on the path and is the vertex just preceeding K and K+1 is the only vertex adjacent to K, then K is deleted from the path. The process then goes back to 1. Shown in Fig. 5.3(b). c) If K+1 is on the path but is not K's immediate neighbour on the path, then let s be the vertex preceding K+1. Then a vertex r on the path between K+1 and K is found out such that r is adjacent to q. Let s be the vertex preceding r. Then the path is modified by deleting edges (q-(K+1)) and (s,r), and adding(q,r) and (K,K+1). So the path becomes ——— q-r ——— K - (K+1) ——— s. So s becomes a terminal point again. Then going back to 1 the next vertex is selected from s. This is shown in Fig. 5.3(c) and (d).

5.6 PROCEDURE

First the personality matrix is read in. Then the column disjoint graph G(V,E) is formed. A column disjoint graph can be represented by a 2-dimensional array or an one dimensional array of list. The array contains as many rows and columns as there are columns in the PLA personality matrix. In other words it is a square array. The elements of the array are boolean number. The (i,j)th element of this array is made true if column c_i is disjoint from c_j else it is made false. After this has been derived a longest path starting from each vertex of G(V,E) is found according to the procedure described above. There will be n paths if there are n vertices. These paths are stored in a list formed by the Pascal data type called pointer.

Now these paths are sorted according to their length. The longest path is selected and is tested for cyclicity of rows. If found that it introduces cyclic condition this path is shortened by deleting an edge. Then all the paths are again sorted according to their lengths. This process is continued till a path is found which does not induce cyclicity in the rows.

The cyclicity can be tested by the following procedure. A direct cyclic graph (DAG) is constructed in such a way that if $r_a > r_b$ then there is a directed edge from r_a to r_b. The DAG can also be represented by a square array, which a row and a column for each row of the personality matrix. Any element (i,j) of this matrix is made true if $r_i > r_i$ else it is made false. After construction of the DAG, each column of it is checked to see if all the elements of the column is false. If such a column say K, is found then all the elements of the K th row is made false. This row, K must be at the top as no other row is above it as indicated by all the false elements of column K of DAG. After this again the same procedure is applied to get the second top most row. This is continued till all rows of the personality matrix are selected or no column can be found with all element false. If the second case occurs then the path intorduces cyclic condition in rows as is evident by the fact that all rows have at least one row above it. This is physically not possible. This method of finding out the order is called topological sort [18].

The following is the column disjoint matrix of the PLA in Fig. 5.1(a).

$$\begin{array}{cccccccc}
 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\
C_1 & F & T & T & T & T & F & F \\
C_2 & T & F & T & F & F & F & T \\
C_3 & F & T & F & F & F & F & F \\
C_4 & T & F & F & F & F & T & T \\
C_5 & T & F & F & F & T & T & T \\
C_6 & F & F & F & T & T & F & T \\
C_7 & F & T & F & T & T & T & F
\end{array}$$

F - False

T - True

The following are the paths found out from each node. The numbers indicate the position of the columns in the personality matrix from left to right. The first number is the starting node.

```
1 4 6 7 2
2 1 5 6 7
3 2 7 5 1 4 6
4 1 2
5 6 7
6 5 7
7 6 5
```

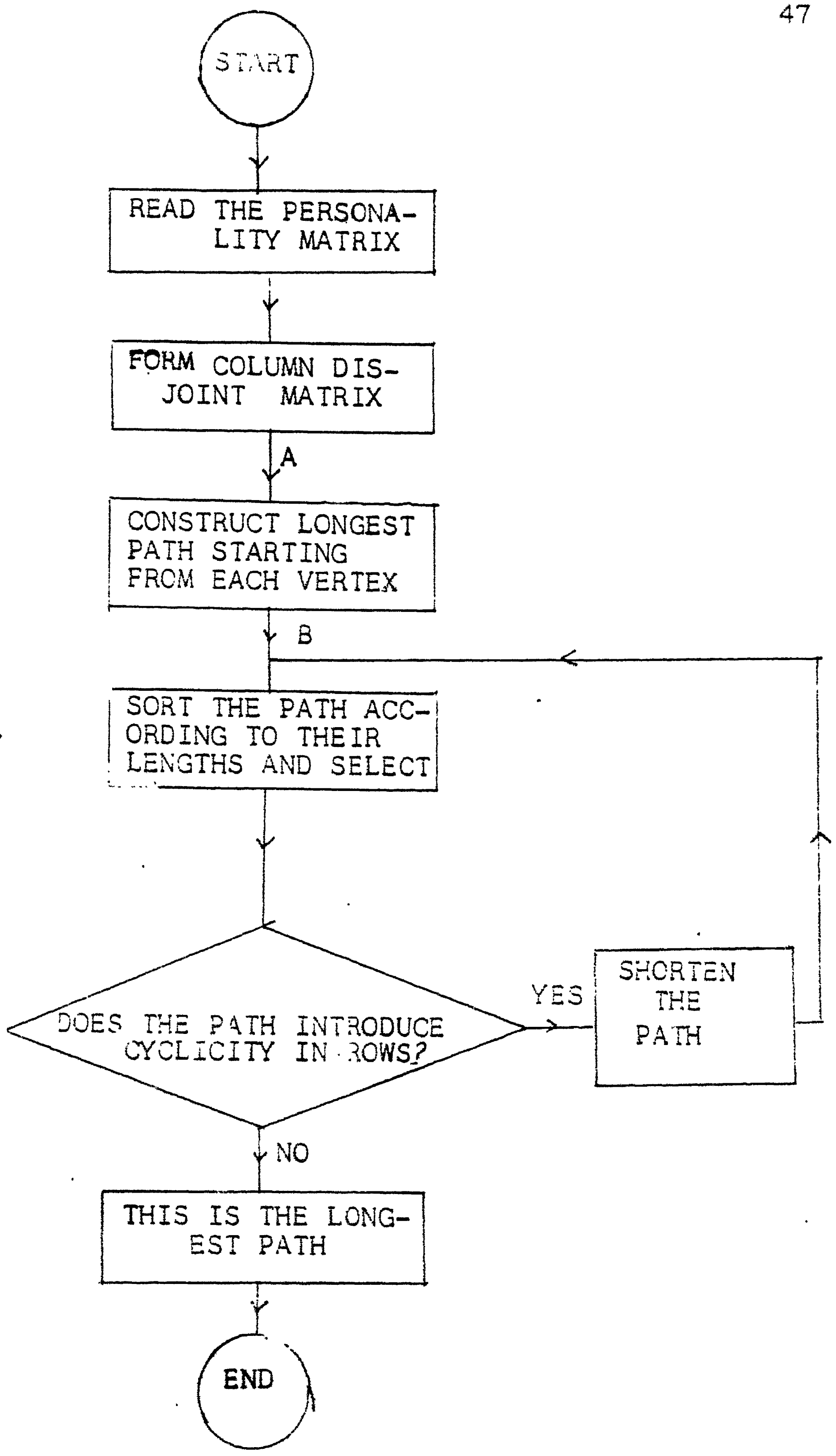
The longest path which do not introduce cyclicity in rows is

3 2 7 5 1 4

The compatible row order is

2 4 1 5 6 3 From top to bottom.

The folded PLA is shown in Fig. 5.1(b).


START
READ THE PERSONALITY MATRIX
FORM COLUMN DISJOINT MATRIX
A
CONSTRUCT LONGEST PATH STARTING FROM EACH VERTEX
B
SORT THE PATH ACCORDING TO THEIR LENGTHS AND SELECT
DOES THE PATH INTRODUCE CYCLICITY IN ROWS?
YES
SHORTEN THE PATH
NO
THIS IS THE LONGEST PATH
END

FLOW DIAGRAM FOR FOLD

CHAPTER 6

STICK DIAGRAM

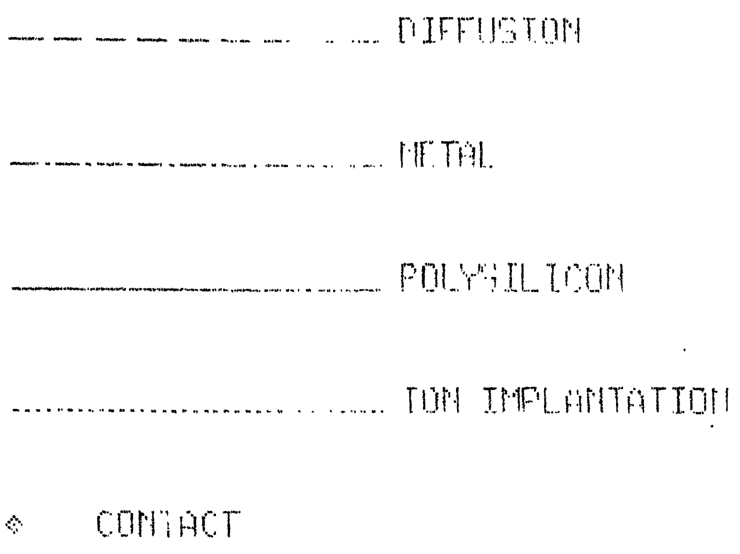
6.1 INTRODUCTION

This chapter describes the stick diagram of a PLA. The colour convention of Mead and Conway [1] is adhered to. They are namely, blue represents metal lines, red represents polysilicon lines, green represents diffusion lines and yellow represents ion-implantation. Dimension (i.e. distance between lines) is not important in stick diagram. Stick diagram is a helpful aid to visualise the layout of a circuit.

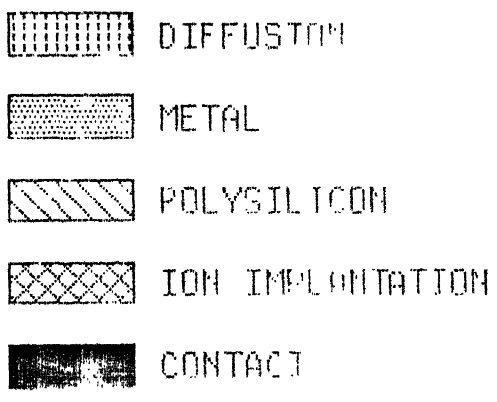
6.2 FOR UNFOLDED PLA

Here first the personality matrix is read in along with number of inputs, number of outputs and number of product terms. First the input pairs (shown in Fig. 6.1) along with the vertical polysilicon lines of the AND plane are drawn. Then the output columns are drawn along with the output inverter and pull-up (Fig. 6.2). Then the horizontal metal lines of AND plane and horzontal polysilicon lines of OR plane are drawn along with the AND plane pull up (Fig. 6.3). Then personalisation diffusion is done. Lastly all the points of same potential are connected and the PLA stick diagram is complete (Fig. 6.4)

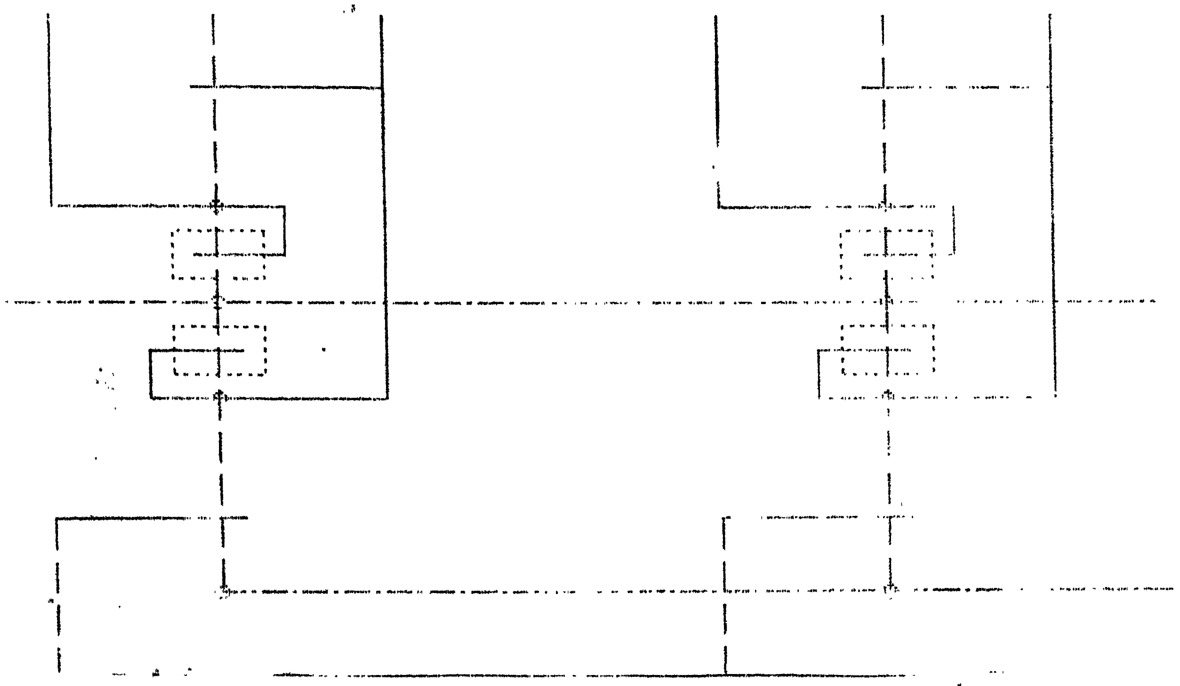
The input pair (Fig. 6.1) consists of two inverters with the drains of pull up connected. Power is supplied at this juncture. The external input goes to the gate of the pull down of the lower inverter. The output from this inverter goes to the AND plane as inverted input. Also this output goes to the



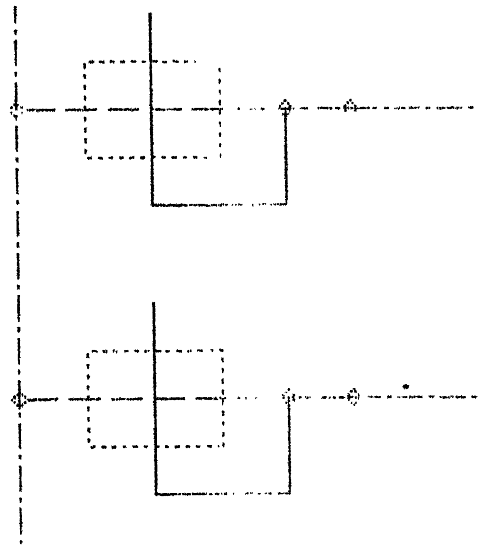
STICK DIAGRAM STYLES



LAYOUT PATTERNS



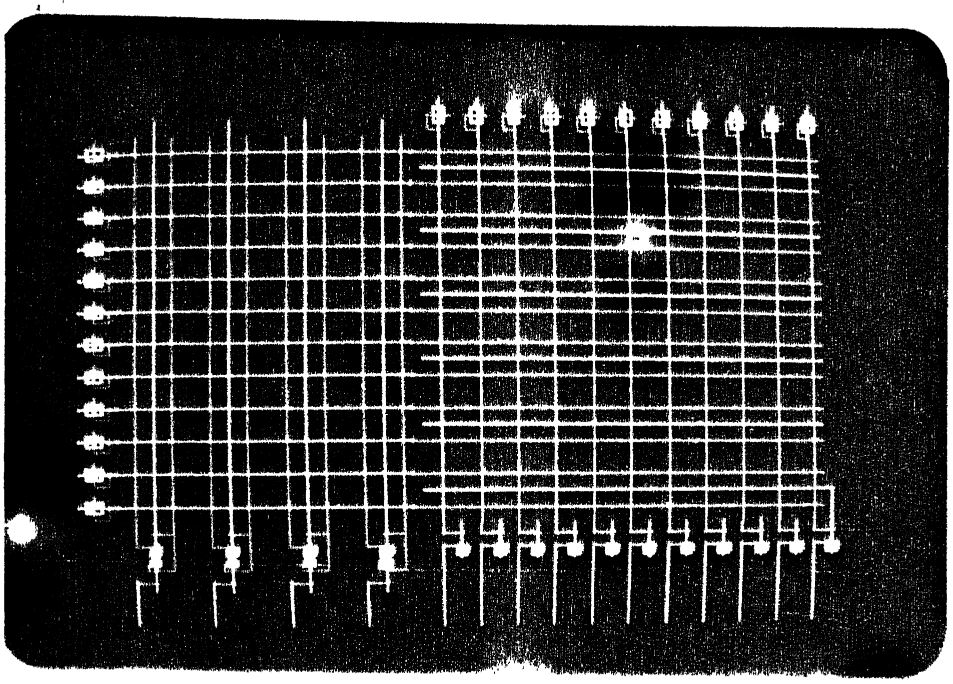
A PAIR OF INPUT PAIRS
FIG 6·1



OR plane pull ups are similar but 90° degree rotated.

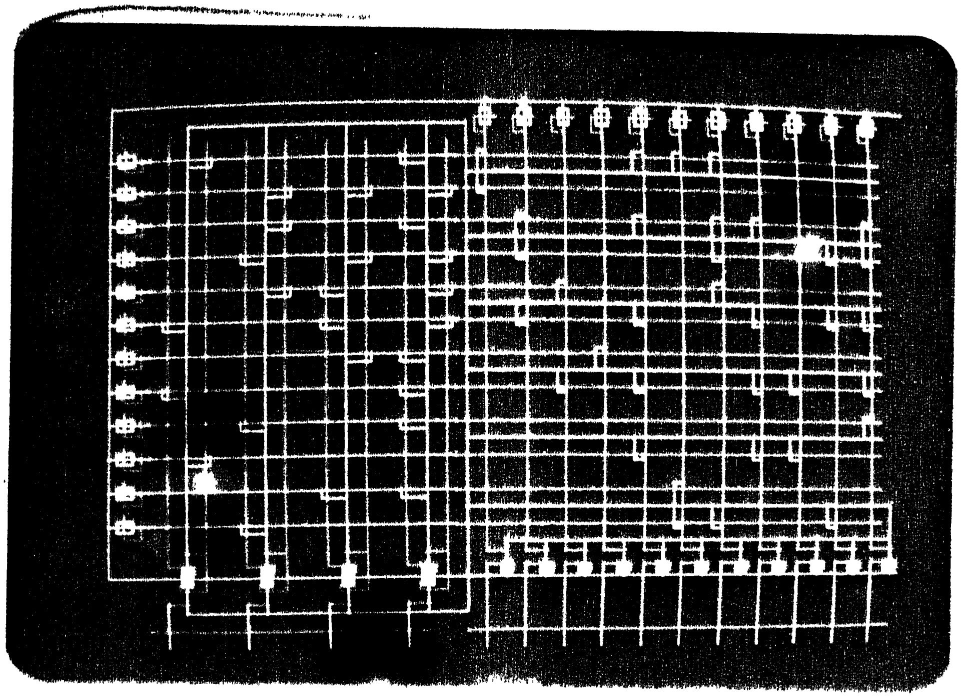
A PAIR OF AND PLANE PULLUPS

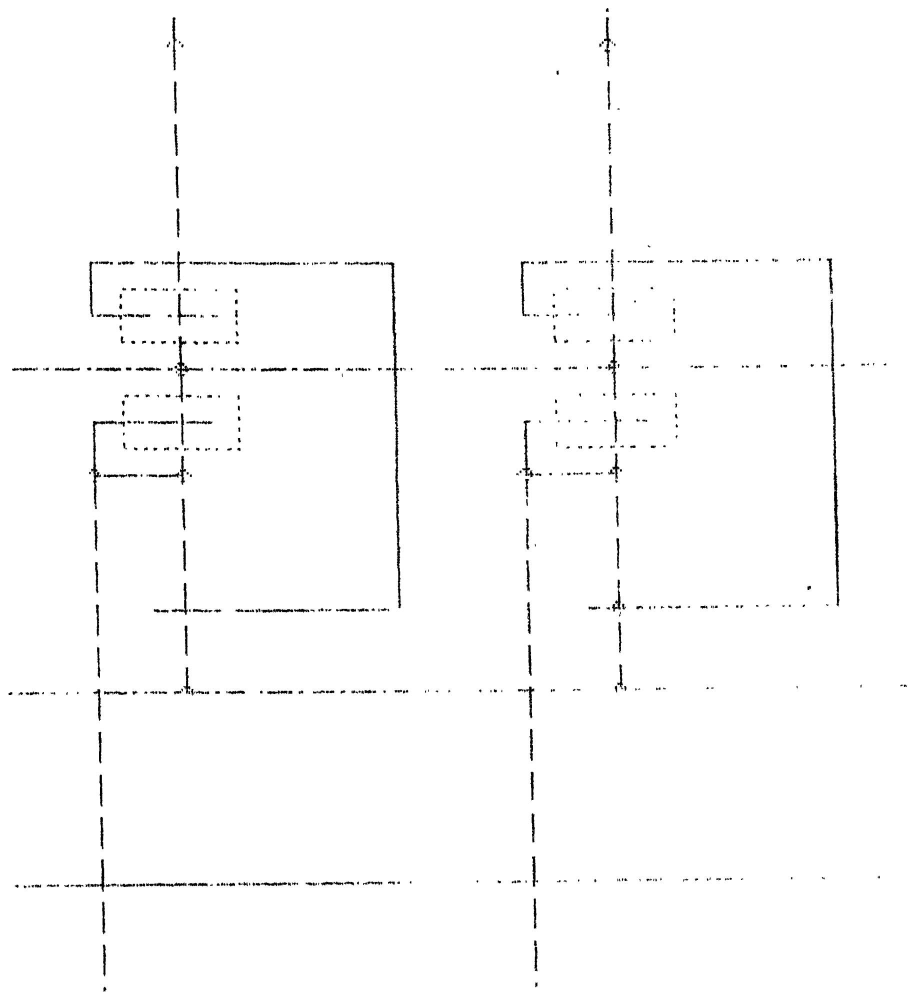
FIG 6·2



Incomplete PLA with Input Pairs,Output Inverters and Pullups.

Fig. 6.3





OUPUT MODULE FOR FOLDED PLA

FIG 6·5

pull down of the top inverter. The output from which goes to the AND plane as the inverted input.

up

The source of the pull/is always connected to a metal line (either AND or OR plane).

6.3 FOLDED PLA

but

It is almost similar to the unfolded PLA/with two differences. Cuts may appear in the AND plane polysilicon lines and in the OR plane metal lines. (Fig. 6.5). The other difference is in the arrangement of output inverters and OR plane pull up. Here both the output inverter and the OR plane pull are brought to the same side of the OR plane. This facilitates folding. Otherwise routing of the power supply lines would have been cumbersome. Here the pull up is above the inverter as shown in the figure. The output from the pull up goes to the input of the inverter.

CHAPTER 7

LAYOUT

7.1 INTRODUCTION

This chapter describes the layout diagram of a PLA. The layout shows the exact arrangement of transistors and interconnection. Here also the colour convention of Mead and Conway is adhered to [1]. Colour convention is necessary to show overlapping of different regions.

7.2 COLOUR CONVENTION

Green	-	represents diffusion region
Red	-	represents polysilicon region
Blue	-	represents metal region
Yellow	-	represents ion-implantation region

Black region surrounded by other region - contact cuts.

The overlapping of regions are indicated by boolean ORing of their colours.

7.3 LAYOUT DESIGN RULES

Here the λ design rules of Mead and Conway [1] are followed. λ represents the highest resolution for photo masking. The design rules are

1. Two diffusion regions must be separated by a distance of at least 3 λ.

2. Two polysilicon regions and two metal region must be separated by a distance at least 2λ ,

3. Parallel polysilicon and diffusion region must be separated by a distance of 2λ ,

4. The minimum dimension allowed for any region is 2 λ ,

5. Contact cut above active area is not allowed,

6. Contact holes must be at least 2λ X2λ in dimension and they must be surrounded by λ thickness of each contacting layer.

7.4 FOR UNFOLDED PLA

For drawing the layout diagram seven different modules are made. Then these modules are placed according to their position.

All these modules are made from a primitive called rectangle It draws a rectangle according to the position of lower left corner, the width and the height of a rectangle and fills its with the specified colour. This is shown in fig. 7.1.

One module is for the pair of input inverters (IP). This is shown in Fig. 7.2. Dimension are also shown there. Here one inverter is above the other inverter with their drains of pull up connected. Power is supplied at this junction.

The pull ups in the AND plane consist another module (PUA). This is a single depletion mode transistor, acting as pull up of the AND plane transistors. This is shown in Fig. 7.3.

Here gate of the transistor is connected to its source.

Fig. 7.4 shows an AND plane module (AC). It consists of two vertical polysilicon lines and one / horizontal metal line. The dimensions are shown in the diagram.

Fig. 7.5 shows the /cell which connects the AND plane and the OR plane (CC). This is another module in the layout diagram. There will be as many connect cells as there are product terms.

One module / consists of an OR cell (OC). This is similar to an AND plane cell, but is rotated by 90 degree

Another cell,(or module) is the OR plane pull ups (OPU). This is similar to AND plane pull ups but is rotated clockwise by 90 degrees.

The last module is for the output inverters (OPC). This is shown in Fig. 7.6. It is an inverter with a depletion mode pull up transistor. The pull up is below the pull down.

Fig. 7.7 depicts how the modules are placed for a three input and three output PLA. After this the points at same potential are connected. For this no module is made.

Before drawing the PLA,personalisation diffusions are drawn. These are nothing but green coloured rectangles.

7.5 FOR FOLDED PLA

For folded PLA the layout diagram is almost similar to that of unfolded PLA. There are two main differences. One is, cuts may appear in the AND plane poly lines and OR plane metal lines. The other is the arrangement of the output inverter and the OR plane pull up. First is indicated in Fig. 7.8. The OR plane pull ups and the output inverters are one the same side of the OR plane. These two together form a module. This module (OM) is shown in Fig. 7.9. The pull up is above the inverter. Its source is connected to an OR plane metal line. The pull up of the inverter is below the pull down.

Other minor differencesare; input pair may appear at top of the AND plane also; so also output inverters; the number of columns is not equal to the sum of input and output.

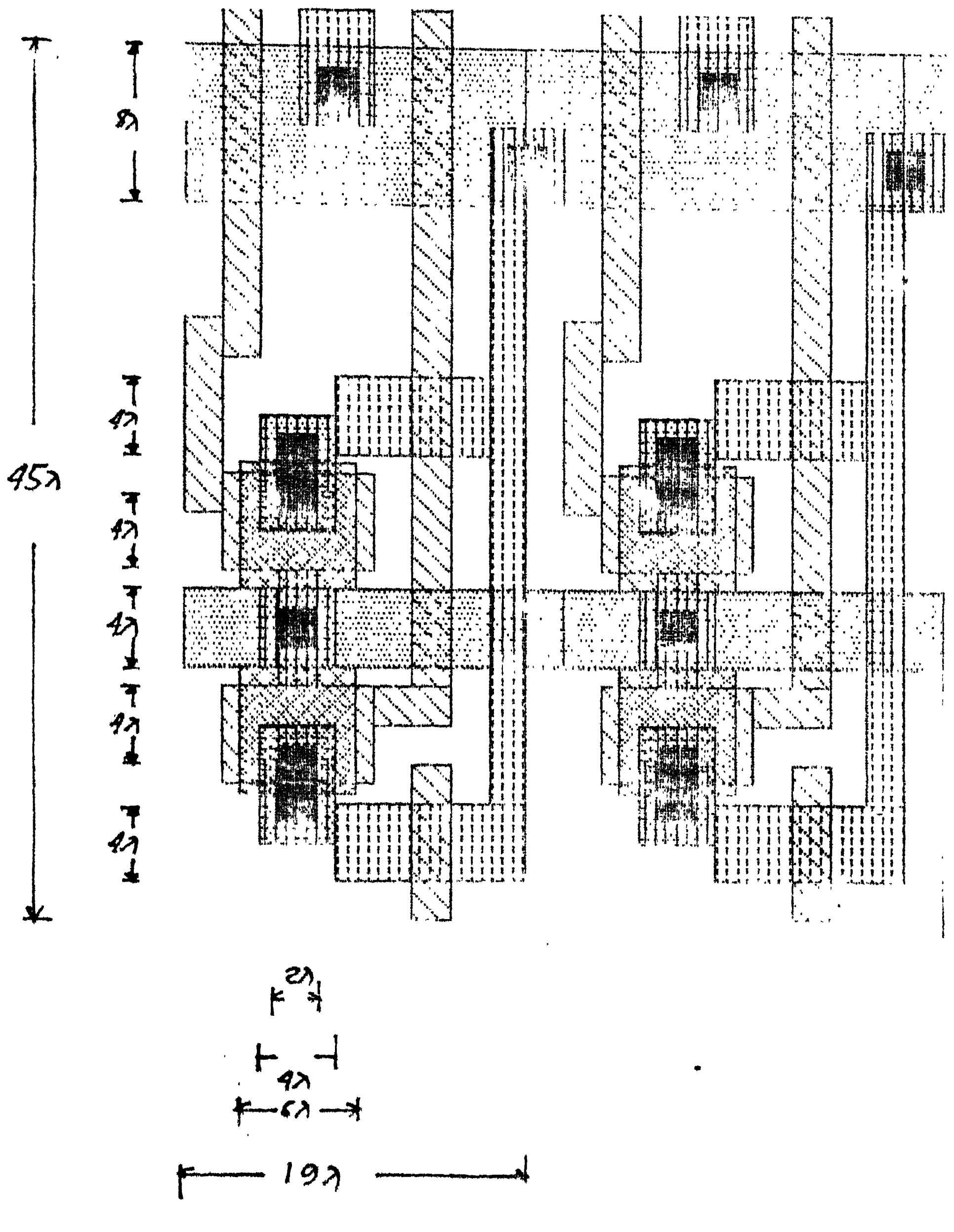
The arrangement for a 3 input and 3 output PLA is shown in Fig. 7.10.

Filled with
required
colour of
shade

(x_o, y_o)

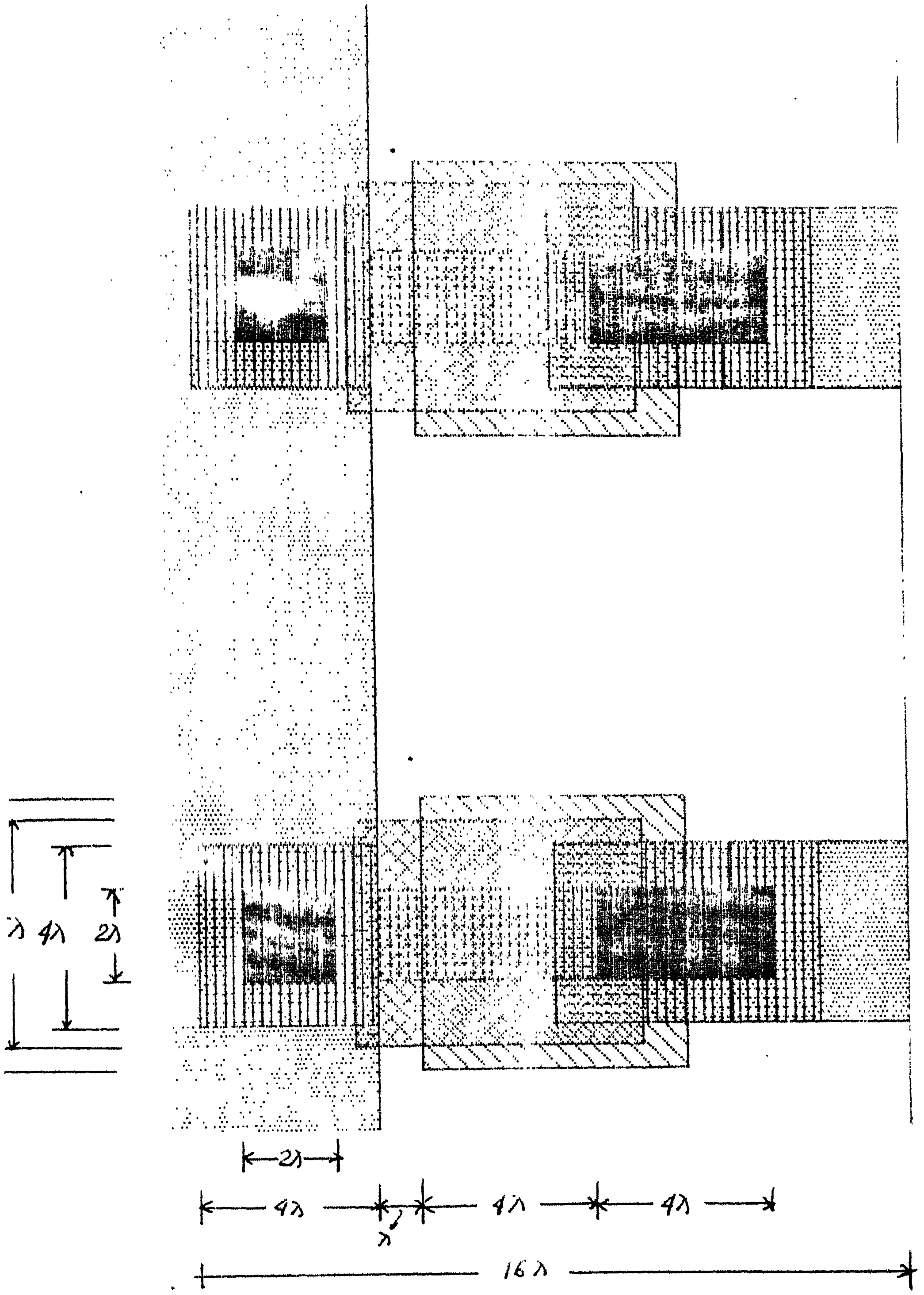
rectang $(x_o, y_o$,w,h,colour code)

Fig. 7.1



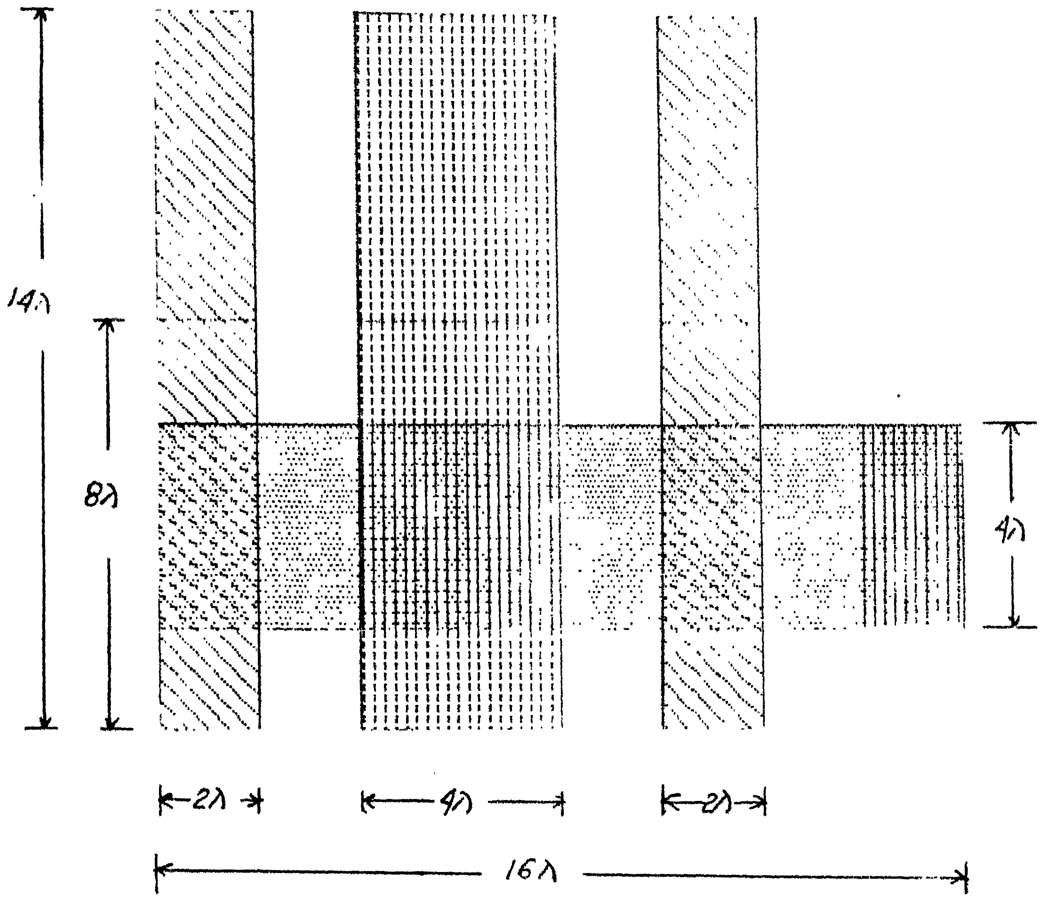
A PAIR OF INPUT PAIRS

FIG 7·2



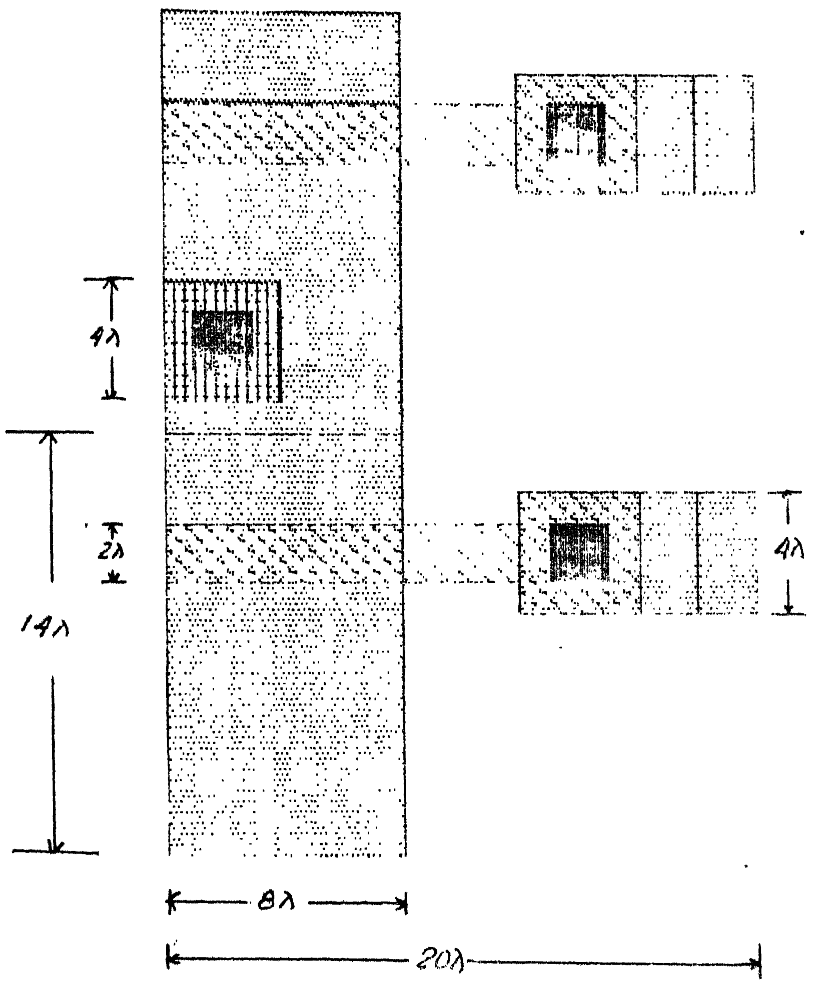
ONE PAIR OF PULLUPS IN AND PLANE

FIG 7·3



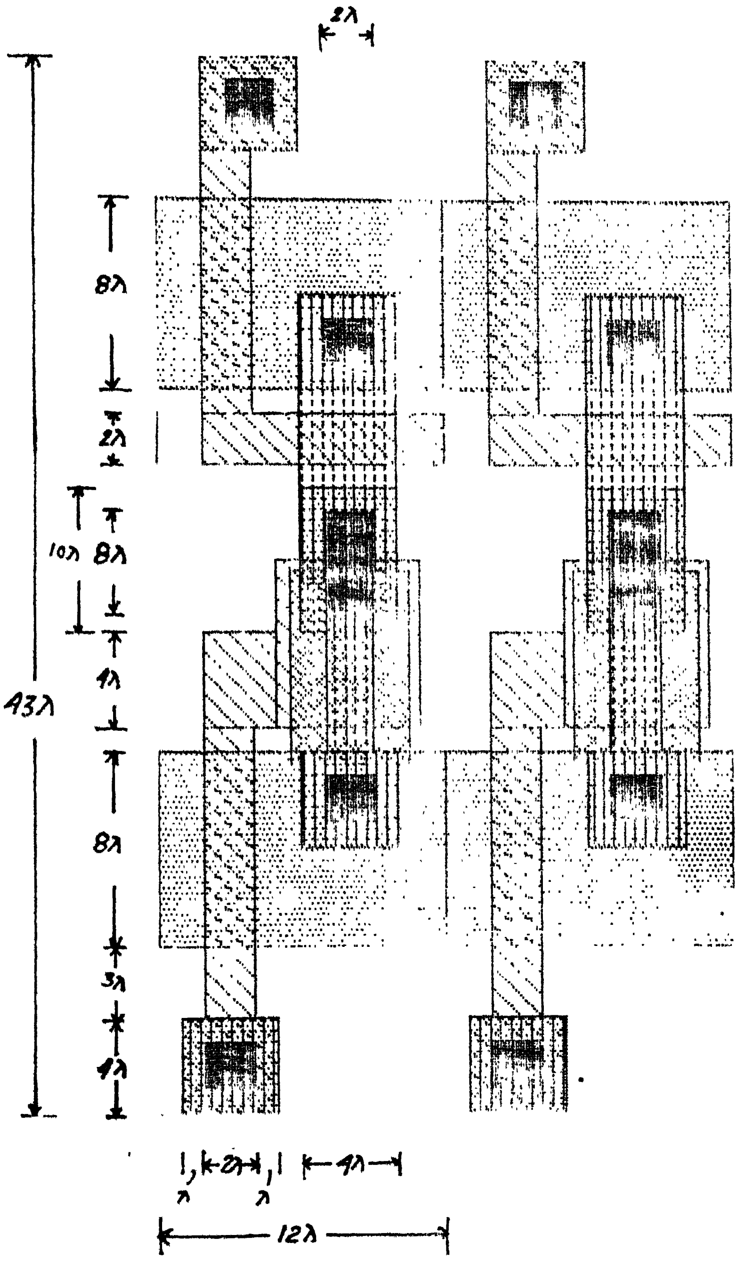
AND PLANE MODULE

FIG 7·4



AND-OR CONNECT CELL·

FIG 7·5



PAIR OF OUTPUT INVERTERS

FIG 7·6

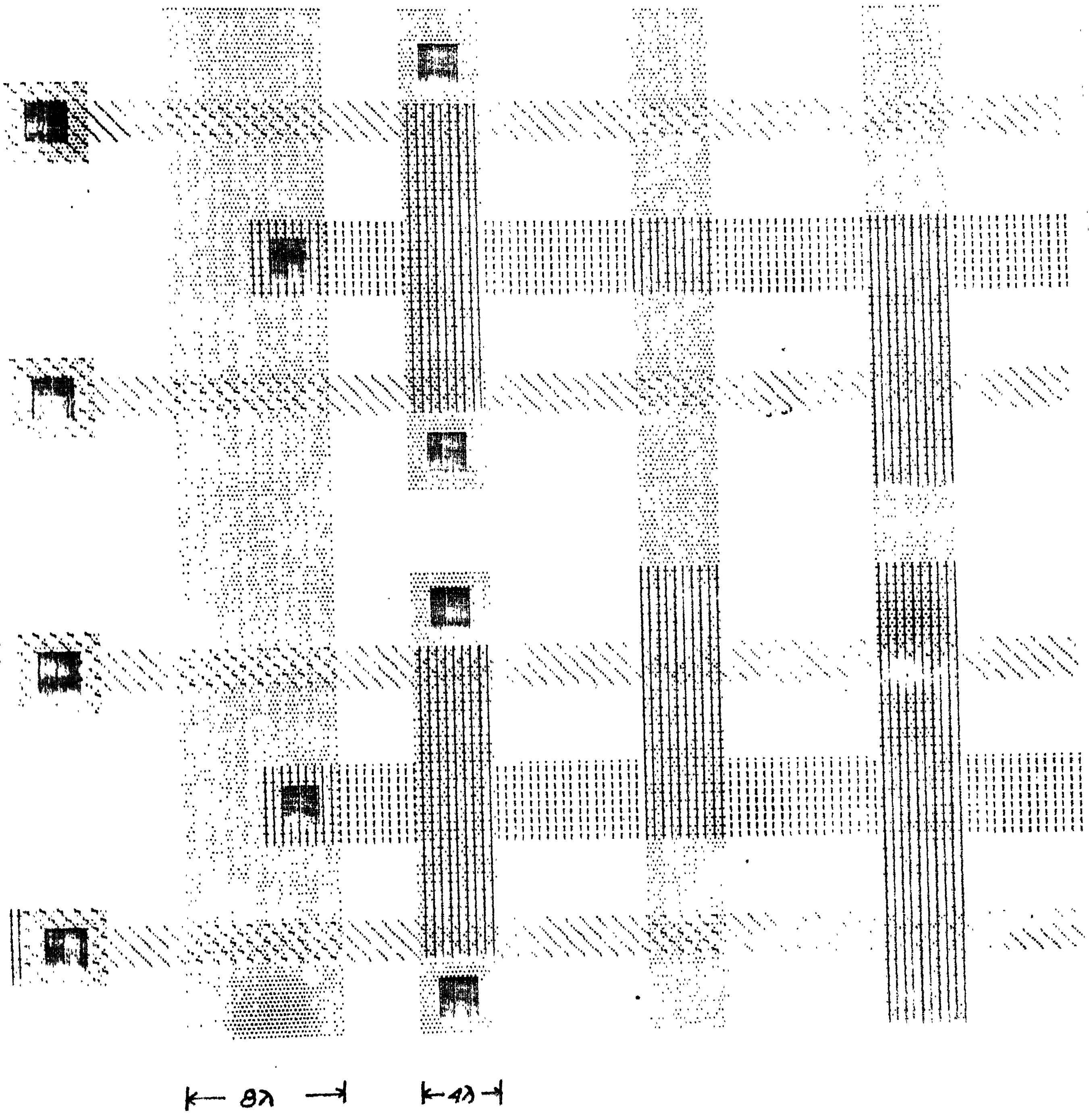
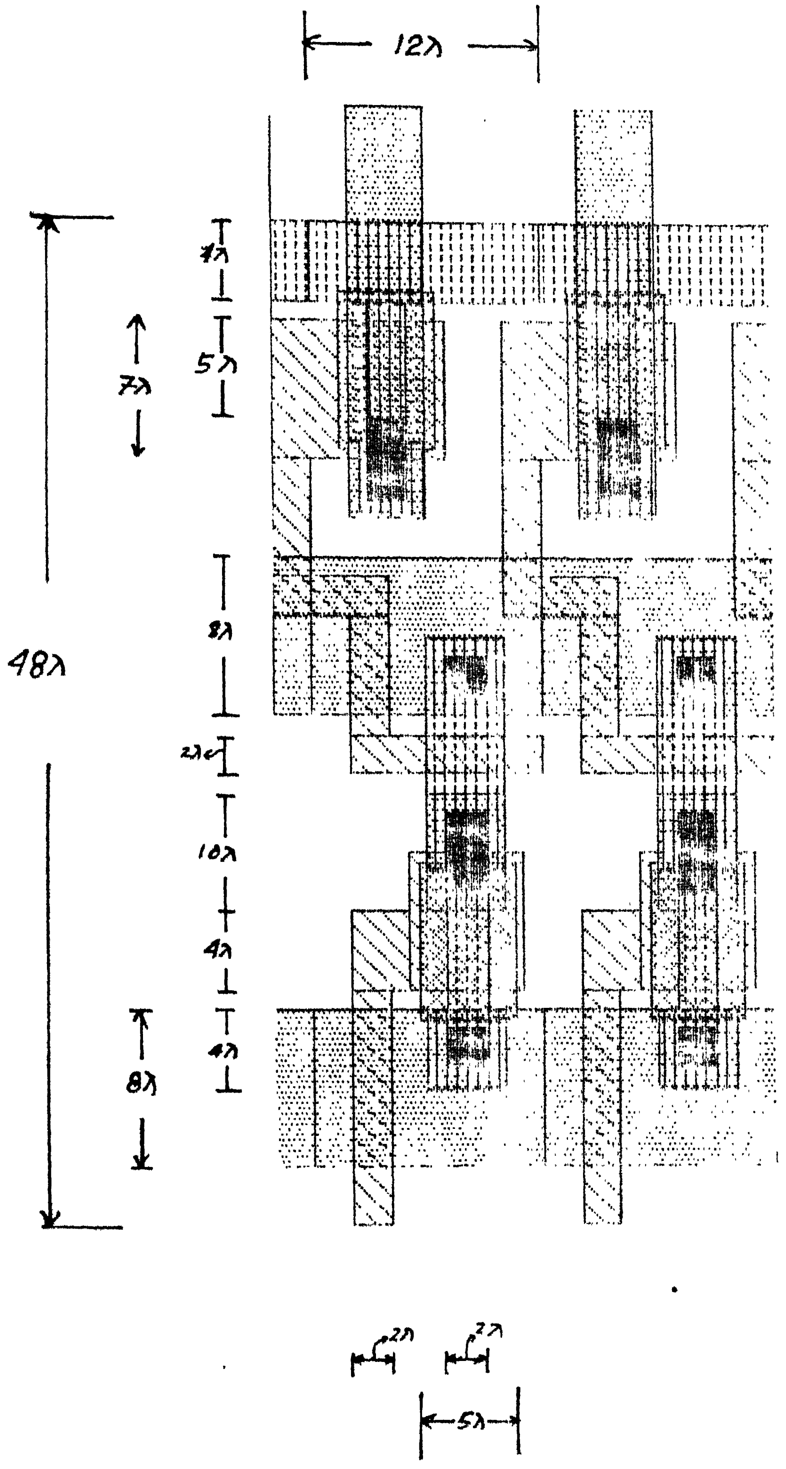
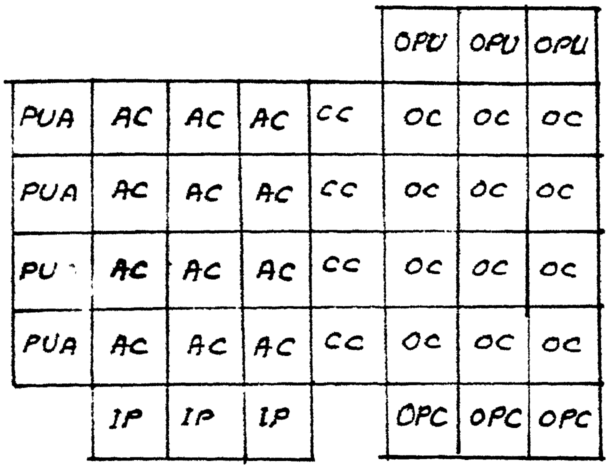


FIG 7·8 A CUT IN THE OR PLANE

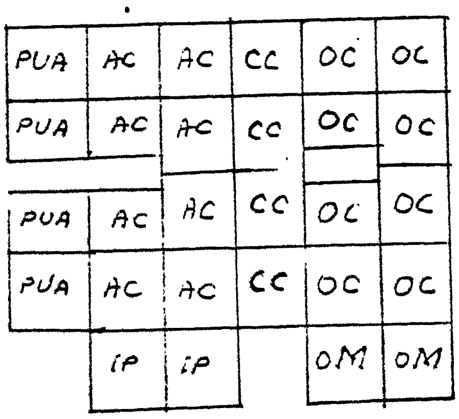


A PAIR OF OUTPUT MODULES FOR FOLDED PLA

FIG 7.9



Arrangement of modules
FIG 7.7



One cut in AND and another in OR plane

Arrangement of modules in a foldes PLA.
FIG 7.10

CHAPTER 8

EXAMPLES

8.1 BINARY TO GRAY CODE CONVERTER

Here number of input is 4 and the number of output is 4. The output functions are expressed in terms of minterms. The input deck for REDUC is shown below (Fig. 8.1(a)).

```
4     4
8 9   10 11 12 13 14 15 -1
4 5    6  7  8  9 10 11 -1
2 3    4  5 10 11 12 13 -1
1 2    5  6  9 10 13 14 -1
```

(a)

```
IXXXIXXX
IOXXXIXX
OIXXXIXX
XIOXXXIX
XOIXXXIX
XXIOXXXI
XXOIXXXI
```

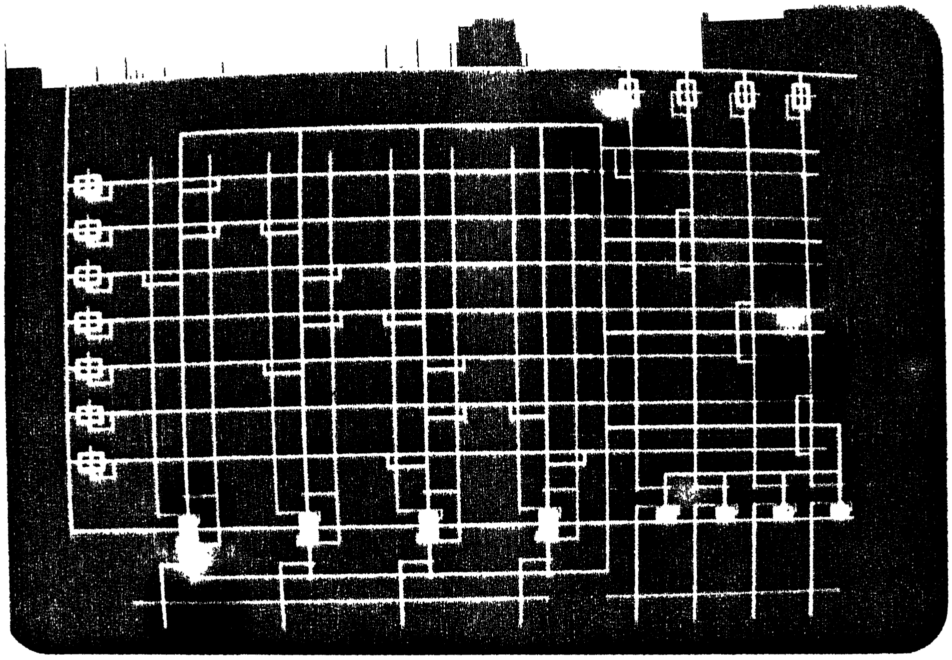
(b)

```
IXXXIXXX
IOXXXIXX
OIXXXIXX
XIOXXXIX
XOIXXXIX
XXIOXXXI
XXOIXXXI
7
3 1 7 5 8 6 4 2
7 6 5 4 3 2 1
```

(c)

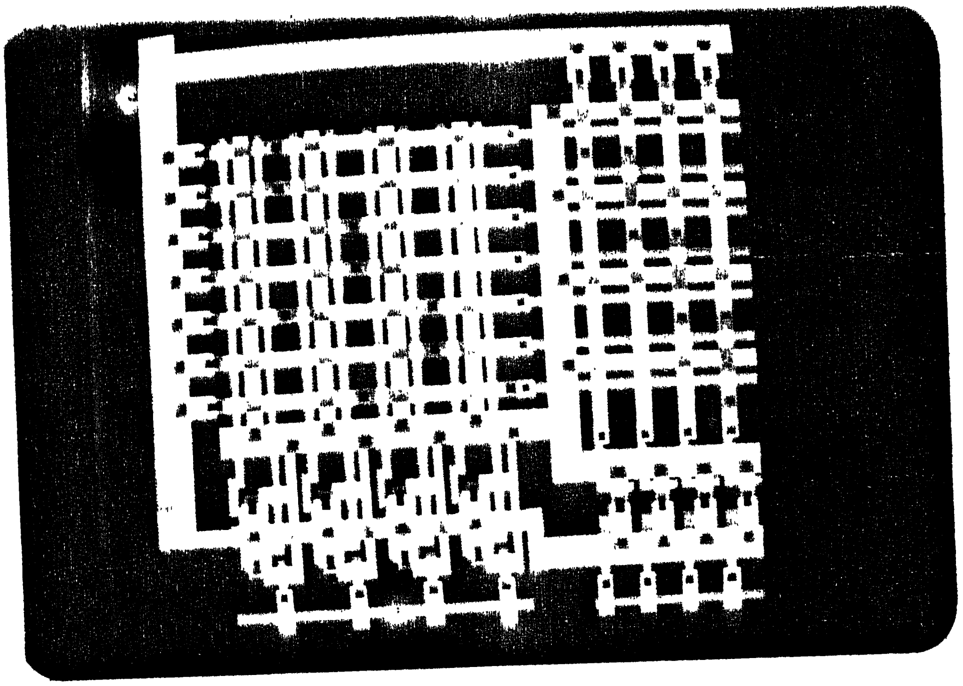
Fig. 8.1

The output from REDUC is the personality matrix shown in Fig.8.1 (b). This in turn is input to FOLD. The output from FOLD is shown in Fig. 8.1(c). The first number is 7. Hence there are 4 foldable pairs. The unreduced stick and layout diagrams are shown in Fig. 8.2(a) and Fig. 8.2(b) respectively. Fig. 8.3(a) and 8.3(b) shows the reduced stick and layout diagram respectively. The folded stick diagram and layout diagram are Fig. 8.4(a) and 8.4(b) respectively.



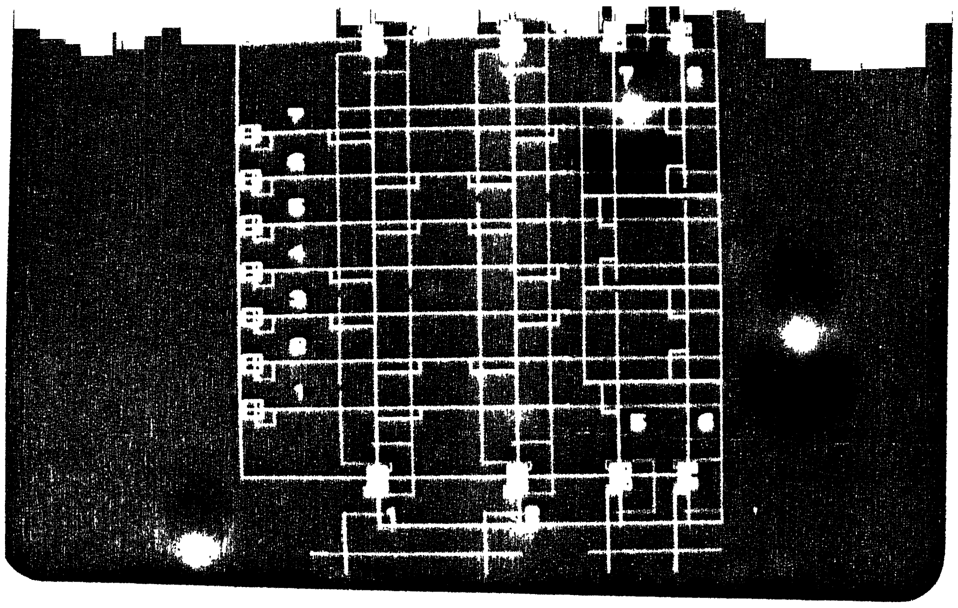
Stick diagram for reduced converter

(a)



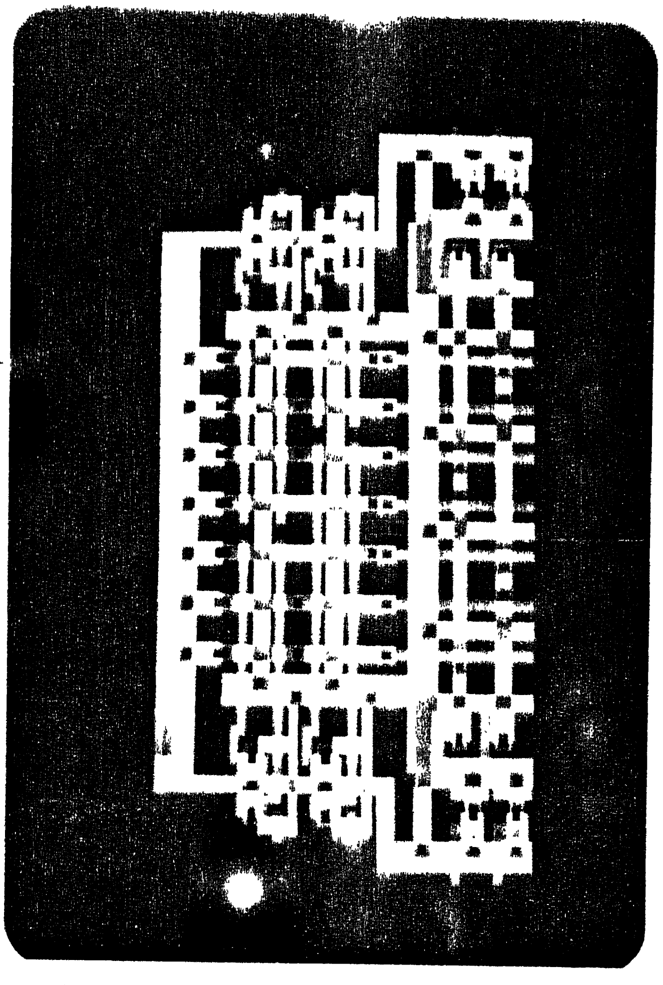
Layout diagram for reduced converter
(b)

[illegible]



Stick diagram for reduced and folded converter

(a)



Layout diagram for reduced and folded converter

(b)

Fig. 8.4 FOLDED CONVERTER

8.2 SEVEN SEGMENT DISPLAY

The expressions for the output functions in the input deck for REDUC are in (Fig. 8.5(a). The number of input here is 4 and number of output here is 7. The first line indicates this.

```
 4  7
0 2  3 5 6 7 8 9 -1
0 1  2 3 4 5 7 8 9 -1
0 1  3 4 5 6 7 8 9 -1
0 2  3 5 6 8 9 -1
0 2  6 8 -1
0 4  5 6 8 9 -1
2 3  4 5 6 8 9 -1
```

(a)

```
     A
F    G    B
E         C
     D
```

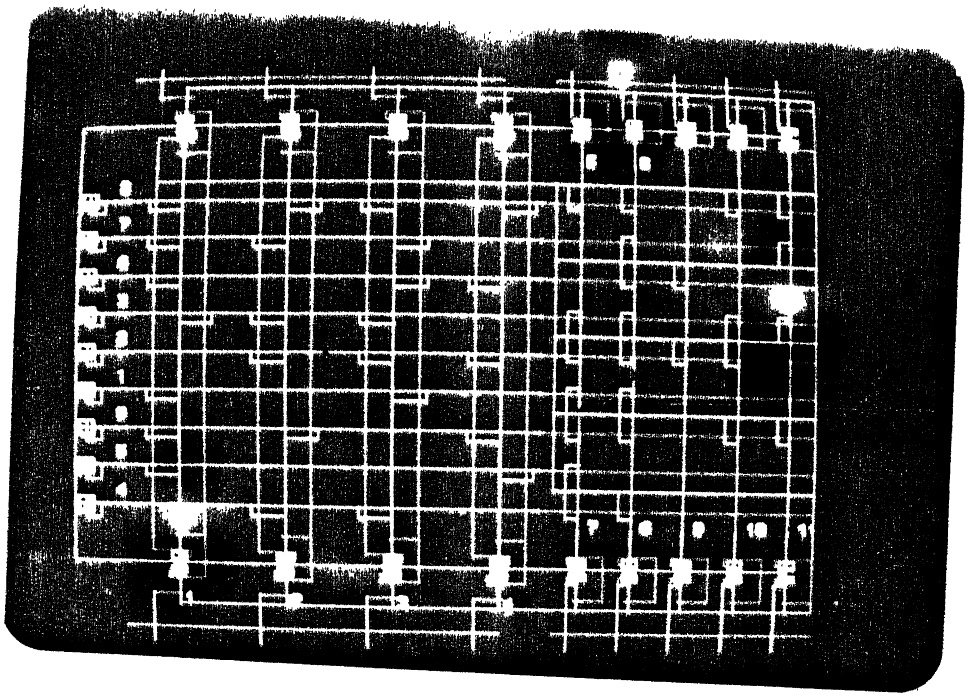
```
OXIXIIXXXXX
XOOOIXXIIIX
IOOXIXXIXII
XOOXXIIXXXX
OXXIXXIXXXX
OXIOXXXIIXX
OOIXXXXIXXI
OIOIIXXIXII
OIXOXIIXXII
3
5 7 8 6
8 7 6 3 2 1 9 5 4
```

(b)

Fig.

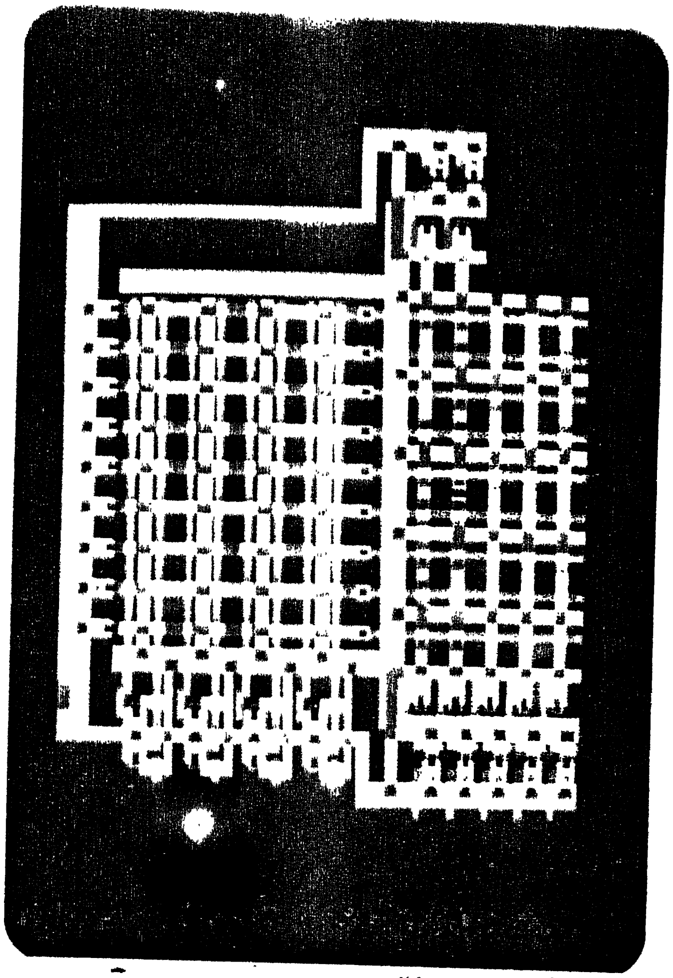
The second line in the input deck specifies A in terms of minterm. The third line specifies B and so on. The output from FOLD is shown in Fig. 8.5(b)(output from REDUC is not shown as it is a subset of the output of FOLD). There are two foldable pairs. This reduces the number of columns from 11 to 9.

The folded stick diagrams and layout diagram are shown in Fig. 8.6(a) and 8.6(b) respectively.



Stick diagram for the PLA for display (reduced and folded)

(a)



Layout diagram for the PLA display (reduced and folded)

(b)

8.3 3 BIT ADDER

As each operand will have three bits the number of input will be six. The number of output is four. Fig. 8.7 shows the input deck for REDUC. The MSB is specified first, then the other bits.

The output from FOLD is shown in Fig. 8.8. The folded stick and layout diagram are shown in Fig. 8.9(a) and Fig. 8.9(b).

8.4 DECADE COUNTER WITH DISPLAY

It has 4 inputs. 4 outputs are for the counter circuit and 7 outputs are for seven segment display. The block diagram of the counter and display is shown in Fig. 8.10(a). Fig. 8.10(b) shows the Karnaugh maps of the outputs. The input deck for REDUC is given below.

```
4 11                                          2nd line - W
7 8 -2 12 13 14 15 11 10 -1                   3rd line - X
3 4 5 6 -2 12 13 14 15 11 10 -1               4th line - Y
1 2 5 6 -2 12 13 15 14 11 10 -1               5th line - Z
0 2 4 6 -2 12 13 14 15 11 10 -1
0 2 3 5 6 7 8 9 -2 10 11 12 13 14 15 -1
0 1 2 3 4 7 8 9 -2 10 11 12 13 14 15 -1
0 1 3 4 5 6 7 8 9 -2 10 11 12 13 14 15 -1
0 2 3 5 6 8 9 -2 10 11 12 13 14 15 -1
0 2 6 8 -2 10 11 12 13 14 15 -1
0 4 5 6 8 9 -2 10 11 12 13 14 15 -1
2 3 4 5 6 8 9 -2 10 11 12 13 14 15 -1
```

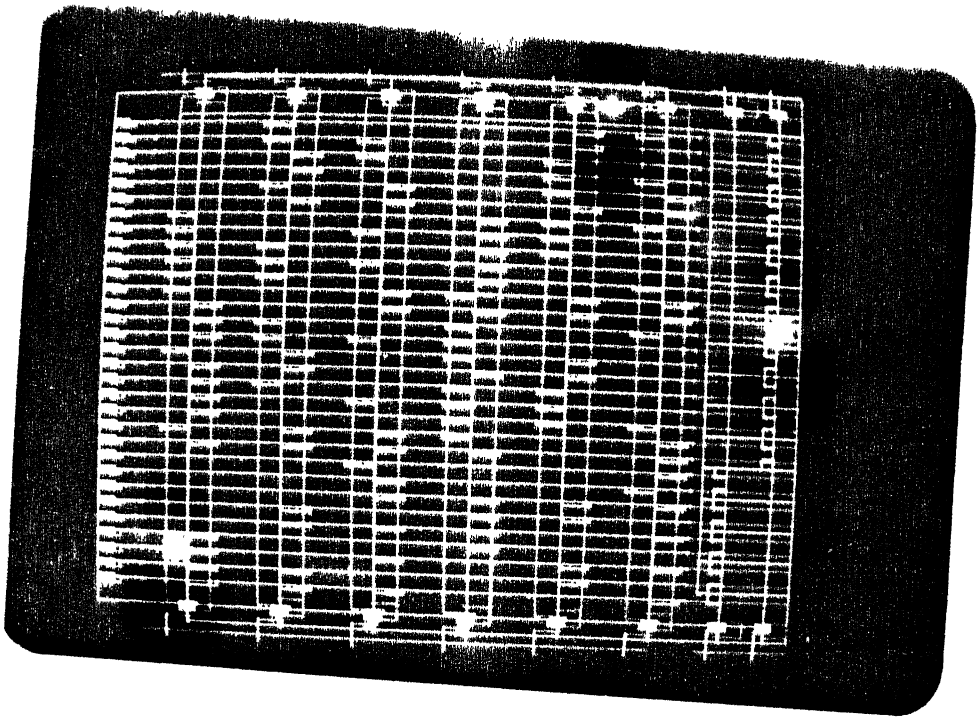
The output from FOLD is shown Fig. 8.11

```
39  40  42  44  46   49   51   53   55   56   58   60   62 -1
 2   3   6   7   9   10   13   14   16   17   20   21   24   27   28   31   34   35  38

39  41  42  45  46   48   49   52   53   56   59   60   63 -1
 4   5   6   7  11   12   13   14   18   19   20   21   25   26   27   28   32   33  34

35  40  41  42  47   48   49   54   55   56   61   62   63 -1
15  22  23  29  30   31   36   37   38   39   43   44   45   46   47   50   51   52  53

54  55  57  56  59   60   61   62   63 -1
```

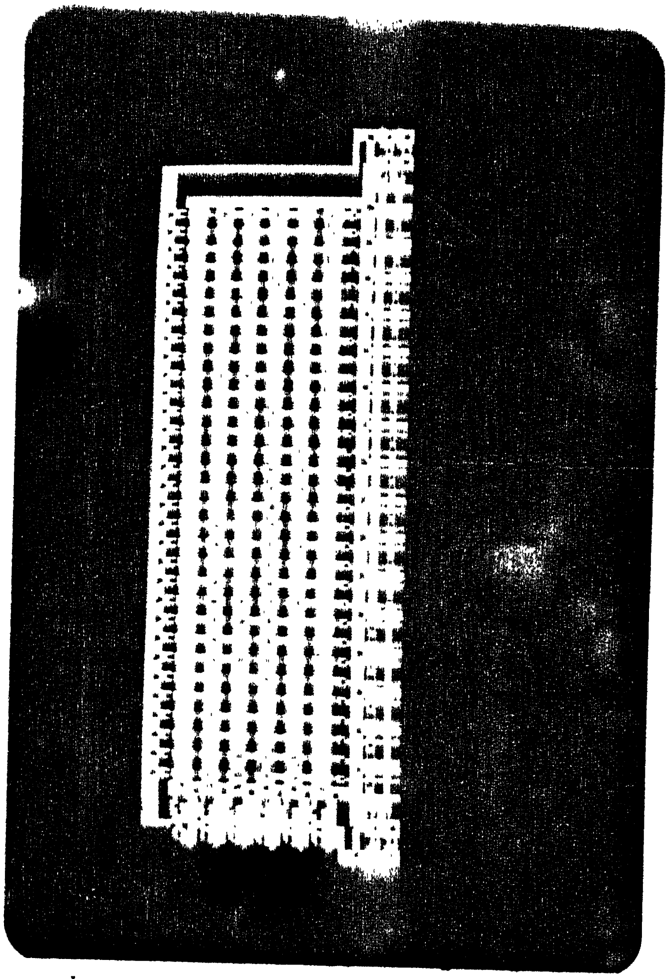
Fig. 8.7

```
XXIXXOIXXX
XXOXXIIXXX
XIIXIIXIXX
XOIXOIXIXX
XIXXOOXIXX
XIOXOXXIXX
XOOXIXXIXX
XOXXIOXIXX
IIIIXIXXIX
OIIOXIXXIX
IXXOOOXXIX
IXOOOXXXIX
IOXOOXXXIX           IXIXIIXXXI
IIXIIXXXIX           IXXIXXXXXI
OIXOIXXXIX           XIIIXIXXXI
IOOOXXXXIX           XIXIIXXXXI
IOXOXOXXIX           XXIIIIXXXI
IXIIIIXXIX             3
OXIOIIXXIX            7 10 8 9
OOOIXXXXIX            8  7 6 5 4 3 24 23 22 21 20 19 18 17 16
OOXIOXXXIX           15 14 13 12 11 10 9 2 1 31 30 29 28 27 26 25
OXOIOXXXIX
OXXIOOXXIX
IIIXXIXXXI
IIXXIXXXXI
```

Fig. 8.8



Stick diagram for 3 bit adder (reduced and folded)
(a)



Layout diagram for 3 bit adder (reduced and folded)

The folded stick diagram along with the D flip flops are shown in Fig. 8.12.

```
              4              12            11

   IXXOIXXXXXXXXXX
   XIIIIXXXIIIXXXX
   XIXOXIXXXXIXXII
   XOIIXIXXIXIIXXI
   XIOIXIXXIXXIXII
   OXOIXXIXXXIXXXX
   XXIOXXIXIXXIIXI
   OXXOXXXIXXXXXXX
   XOXOXXXXIXXIIXX
    IXXXXXXXXXXXXXI
   XXOOXXXXXIIXXIX
   XOXXXXXXXIXXXXX
     5


    1   2   6  13   8  12
   10   8   6   1   5   4   3   9   7  12  11   2
```

Fig. 8.11

The number in Fig. 8,12 indicates position of a row from top and position of a column from left in the personality matrix of Fig.8.11.

	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	1	0	X	X

W^{n+1}

	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	X	X	X	X
10	0	0	X	X

X^{n+1}

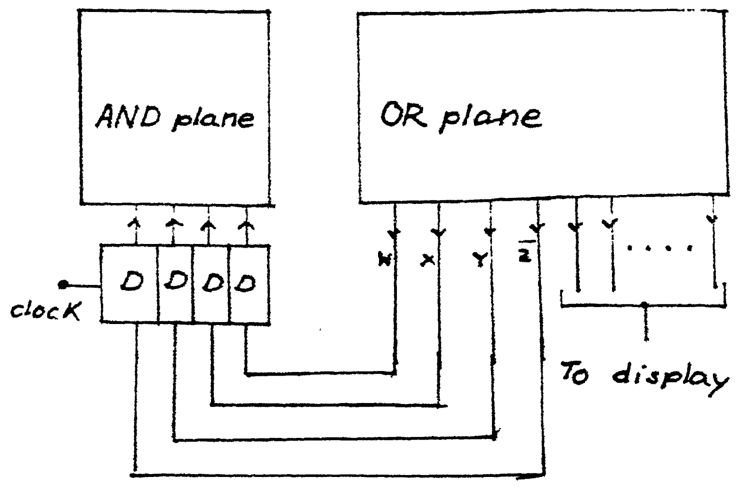
^{n} \ Y^nZ^n	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	X	X	X	X
10	0	·	X	X

Y^{n+1}

(b)

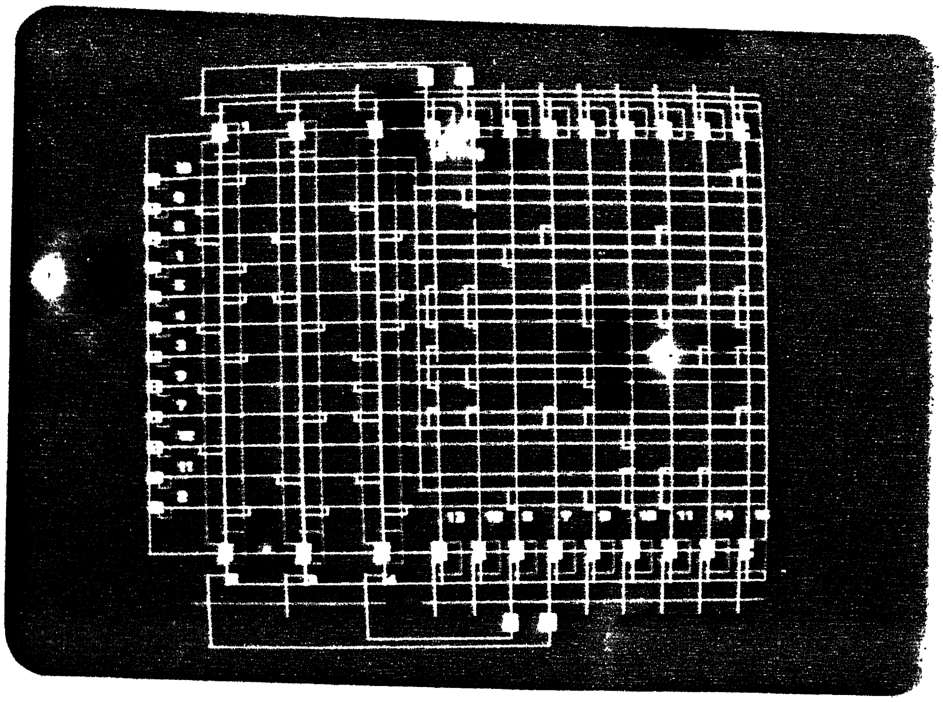
W^nX^n \ Y^nZ^n	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

Z^{n+1}



(a)

Fig. 8.10 Decade Counter With Display.



Stick diagram for the decade counter with display

Fig. 8.12 Stick Diagram for Counter

8.5 CONCLUSION

In this work, design of Programmable Logic Arrays macros have been considered, including logic minimization and folding. Some examples have been shown in Chapter 8. Many other PLAs have also been generated. The minimization and folding programs are found adequate for moderately large PLAs. The individual modules have been tested by SPICE. Also one example has been tested. [illegible] SPICE result was found satisfactory. Though sufficient for moderately large PLAs, PLAs with large number of inputs may not be reduced by REDUC. FOLD was able to fold a 50 column PLA.

The folding program can only fold columns. If row folding is also added then it can be more effective. Also minimization and folding can be combined to obtain optimal reduction in area. Adding a code generator for error detection can take PLAGE one step nearer to total automation in design of PLA. Also a PLA partitioner can be added, as it has been seen that in same cases partitioning a PLA into two parts can reduce area more.

APPENDIX A

USERS' MANUAL FOR PLAGE

The general relation between the programs is described in Chapter 3. All together there are seven programs in PLAGE. They are connected by files.

To use PLAGE, the user will have to input data to either REDUCE or DIRECT.

DIRECT is a program which processes the input data to make them suitable for other programs connected to it. Any file with extension INP can be its input file. This data file can be prepared with the program editor (PED) of the ND system. The format of the data file is given below.

1st line	no. of inputs(i) no. of product terms(p) no. of outputs(O)
2nd line	names of the input variables (must be capital letters,
3rd line	Product term expressed in term of input variables (inverse of an input variable is shown by an apostrophe following the variable name).
4th line	same as 3rd line
.	
.	
.	
(p+2)th line	same as 3rd line, for the last product term.
(p+3)th line	1st output, shown by the serial number of the product terms. The serial number of the product term expressed in the 3rd line is 1, that expressed in the 4th line is 2 and so on.

(p+4)th line same as the above, but for the second output.
(p+0)th line same as (p+3)th line but for the 0th output.

The following is an example of an input file for the program DIRECT. It describes a full adder circuit where A,B and C are the inputs. S stands for the output sum and C stands for the carry output.

$$S = ABC + A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C$$

$$C = AB + AC + BC$$

```
3    7    2
 ABC
 ABC
 AB'C'
 A'BC'
 A'B'C
 ABC
 AB
 AC
 BC
 1  2  3  4
 5  6  7
```

REDUC is a combinational logic minimization program. Its input is any file with the extension INP. Here a function is defined in terms of the decimal expression of the minterms. The first line of the file contains the number of input the number of output. Then each function is defined in terms of the minterms. Each function definition must be terminated by -1. Don't care terms can also be added. Don't care terms are defined after the minterms. In this case the conclusion of minterm is indicated by -2. The end of the don't care terms is indicated by -1. The following K-maps defines a boolean function with 4 inputs and two outputs.

	00	01	11	10
00	1	0	0	1
01	0	1	1	Ø
11	0	1	1	Ø
10	1	0	0	1

f_1

	00	01	11	10
00	0	1	1	0
01	1	1	Ø	1
11	0	Ø	1	0
10	0	Ø	1	0

f_2

Here the input file will look as follows

```
4  2
0  2  5  7  13  15   8  10  -2  6  14  -1
1  3  4  5   6  15  11  -2   7 13   9  -1
```

Both programs REDUC and DIRECT have the same output file called REDOUT:OUT. For the example cited above the output will look like

```
3       7       2
IOOIX
OIOIX
OOIIX
IIIII
IIXXI
IXIXI
XIIXI
```

The above matrix is the personality matrix of the PLA of the boolean function. REDUC also outputs to another file called REDFI:OUT. It gives the detailed description of the boolean function, before reduction and after reduction. This file is not necessary for inter-program communication. But it is essential for the proper execution of REDUC, which unless will show runtime error.

FOLD is a program for column folding of the PLA. The file REDOUT:OUT acts as its input file. Hence to run this program the user does not have to input any extra data. This program has two output files, FOLOUT:OUT and OPFILE:OUT. FOLOUT:OUT communicates with stick diagram and layout programs. OPFILE:OUT is a file like REDOUT:OUT, it gives a detailed description of the folding process. It is not necessary for inter program communication but is essential for proper running of FOLD. FOLOUT acts as an input file to STKF and LAYF. The format of FOLOUT is same as REDOUT but with three extra lines.

The first line contains an integer. This number gives the path length, The second line gives the pairs that can be folded. This line can be manipulated to change the order of the columns in the stick diagram or layout diagram. The last line gives the order of rows from top to bottom.

The programs STK and LAY takes input data from file REDOUT:OUI They draw the unfolded PLA. Programs STKF and LAYF reads data from FOLOUT:OUT and they draw the foled output.

APPENDIX B

SPICE SIMULATION

B.1 SIMULATION

Simulation is an integral part of every design. Before any design is made into reality it has to be checked from every possible point of view. Only after getting satisfactory results from simulation the design is implemented.

Programmable logic Array, as the name suggests, does not have a fixed circuit. Hence general simulation is not possible. So an example is taken for simulation. The example choosen here is the folded Binary / to Gray Code Converter. Its layout and stick is shown elsewhere.

B.2 THE EQUIVALENT CIRCUIT

Fig. B.1 shows the equivalent circuit of the lower portion of the PLA. Here several subcircuits are made. One subcircuit is for the input pairs (IP). For the output inverters and OR plane pull up a subcircuit is made. The following is a description of each subcircuit.

1. IP : - For input pairs, it contains another subcircuit NIP1, it is for one inverter.

C1	-	gate to ground capacitance of pull down
C2	-	gate to ground capacitance of pull up
C3	-	gate to power line capacitance of pull up.
R1	-	resistance of polysilicon from no inverted

R2 - resistance of polysilicon from inverted input to AND plane.

2. ANDPR :- For the part of AND plane which do not depend on personalisation.

C1 - horizontal metal to vertical polysilicon capacitance

R1 - resistance along polysilicon to next AND plane cell

3. ANDDIF :- For personalisation transistor in the AND plane.

C1 - capacitance between vertical poly lines and horizontal metal lines.

C2 - capacitance between metal line and ground.

4. NOADIF :- For AND plane cell when there is no personalisation diffusion.

C1 and C2 same as that of ANDDIF but not of the same value.

5. OP :- Represents the output inverter along with OR plane pull up.

C4 - capacitance between gate of the pull down of the inverter to ground.

C5 - capacitance between gate of pull up and power line.

C6 - capacitance between gate of pull up of inverter and ground.

C7 - capacitance between OR plane pull up gate and ground.

C7D - capacitance between OR plane pull up gate and power line.

R3 - resistance of polysilicon from gate of OR plane pull up and inverter pull down

R2 - resistance between external output to gate of inverter pull up.

6. ANDPU - For AND Plane pull up.

C1 - capacitance between gate and ground

DRDIF, NORDIF and ORPR are similar to ANDDIF, NOADIF and ANDPR respectively.

B.3 OUTPUT

Here only the most significant bit of the input (1) is changed and the change in output noticed. Others bits in the input are held low. The output should be like the following

Node No.	1	41	40	38	39
should be	low	low	low	low	low
SPICE output	OV	0.1588V	0.1588V	0.1588V	0.1588

continued

Node No.	1	41	40	38	39
should be	high	low	low	high	high
SPICE output	OV	OV	OV	4.994V	4.994V

Fig. B.2 shows the transient response at node 39 for the pulse at input (node 1).

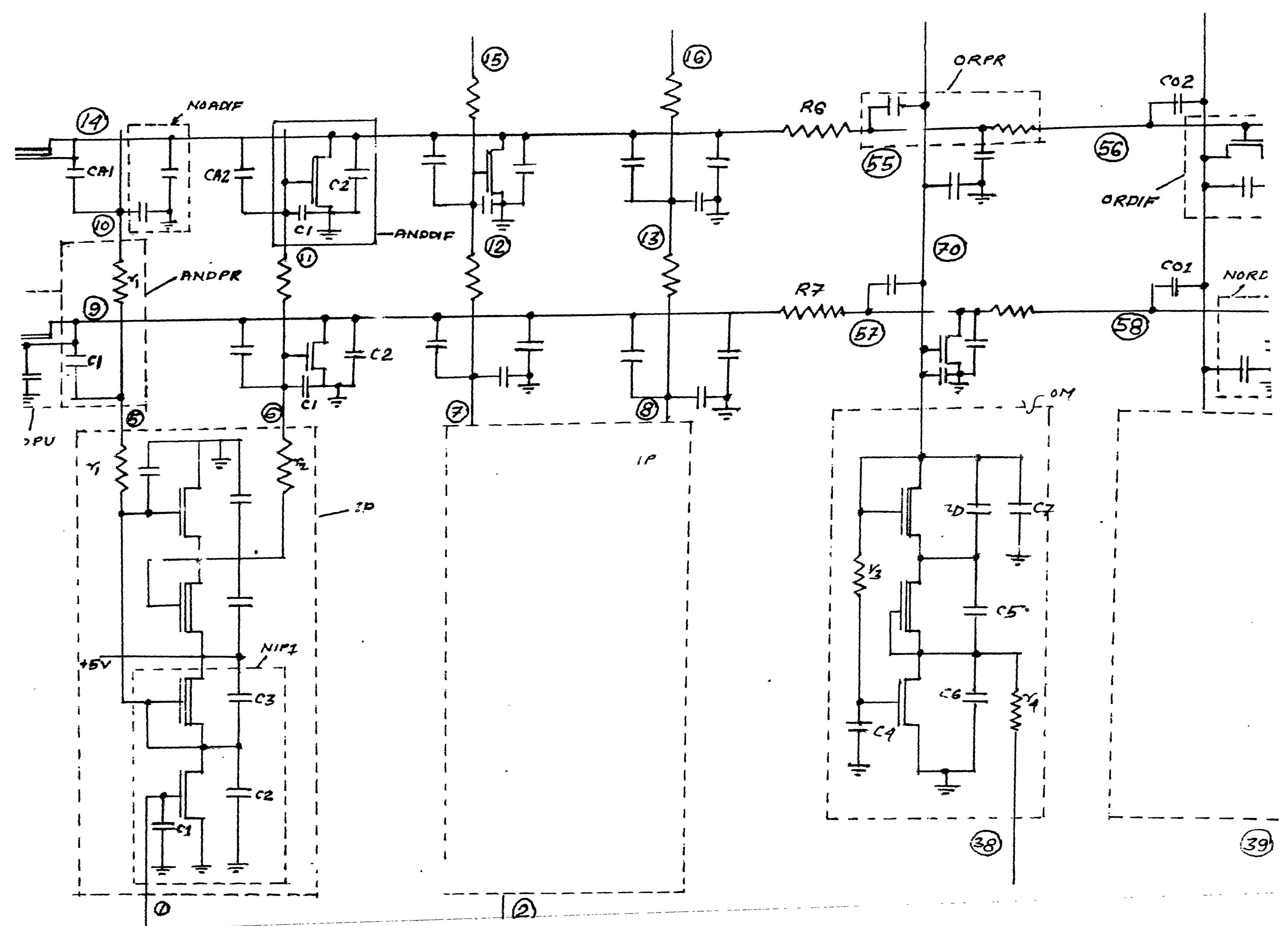
From this, the propagation delay time is = 136 nSec.

Fig. B.2 shows the dc transfer characteristic.

B.4 SPICE INPUT DECK

All devices and parasite parameters are taken from [20].

```
PLA-BINARY TO GRAY

X1   1 5 6 100 IP
X2   2 7 8 100 IP
X3   3 34 35 100 IP
X4   4 36 37 100 IP
XPU1  9 100 ANDPU
XPU2 14 100 ANDPU
XPU3 17 100 ANDPU
XPU4 20 100 ANDPU
XPU5 23 100 ANDPU
XPU6 28 100 ANDPU
XPU7 33 100 ANDPU
XOP1 70 38 100 OP
XOP2 59 39 100 OP
XOP3 42 40 100 OP
XOP4 43 41 100 OP
XANDPR1 9 10 5 ANDPR
```



14
CA1
NORDIF
10
CA2
ANDPR
9
C1
PU
5
ANDDIF
11
C2
C1
6
ID
NIP1
+5V
C3
0
15
12
7
16
13
8
IP
2
R6
R7
ORPR
55
57
70
56
58
ORDIF
CO2
CO1
NORC
OM
C7
C5
C6
C4
r3
r4
38
39

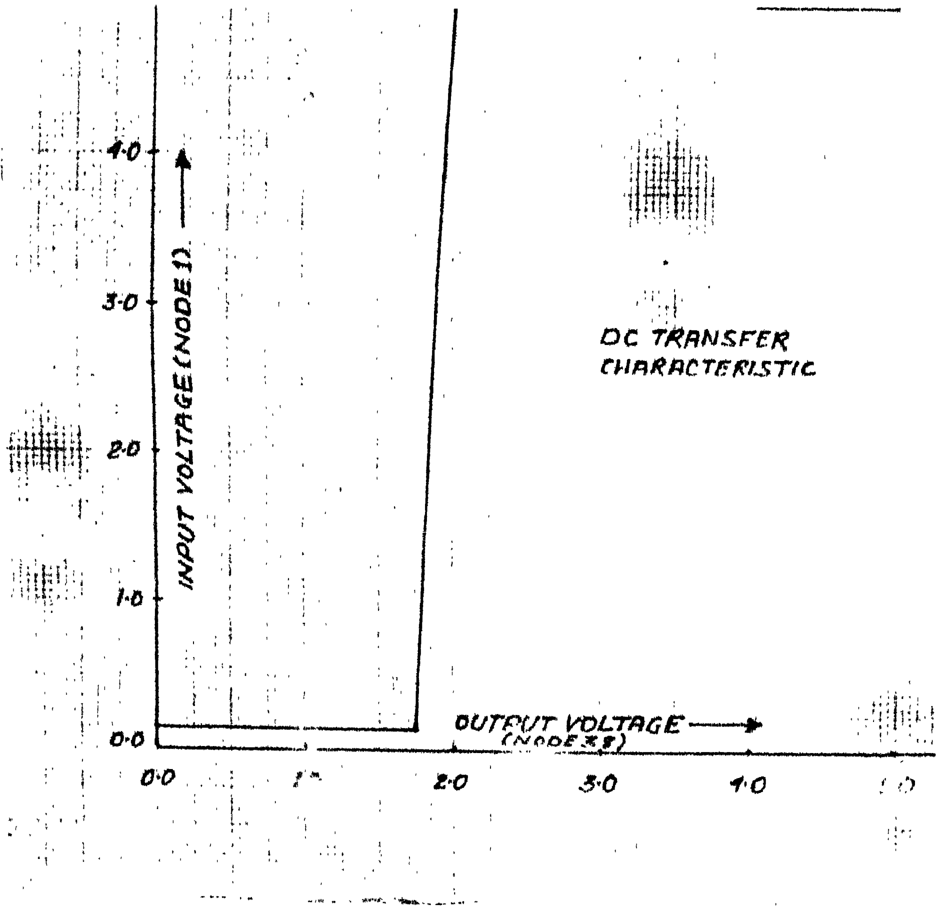
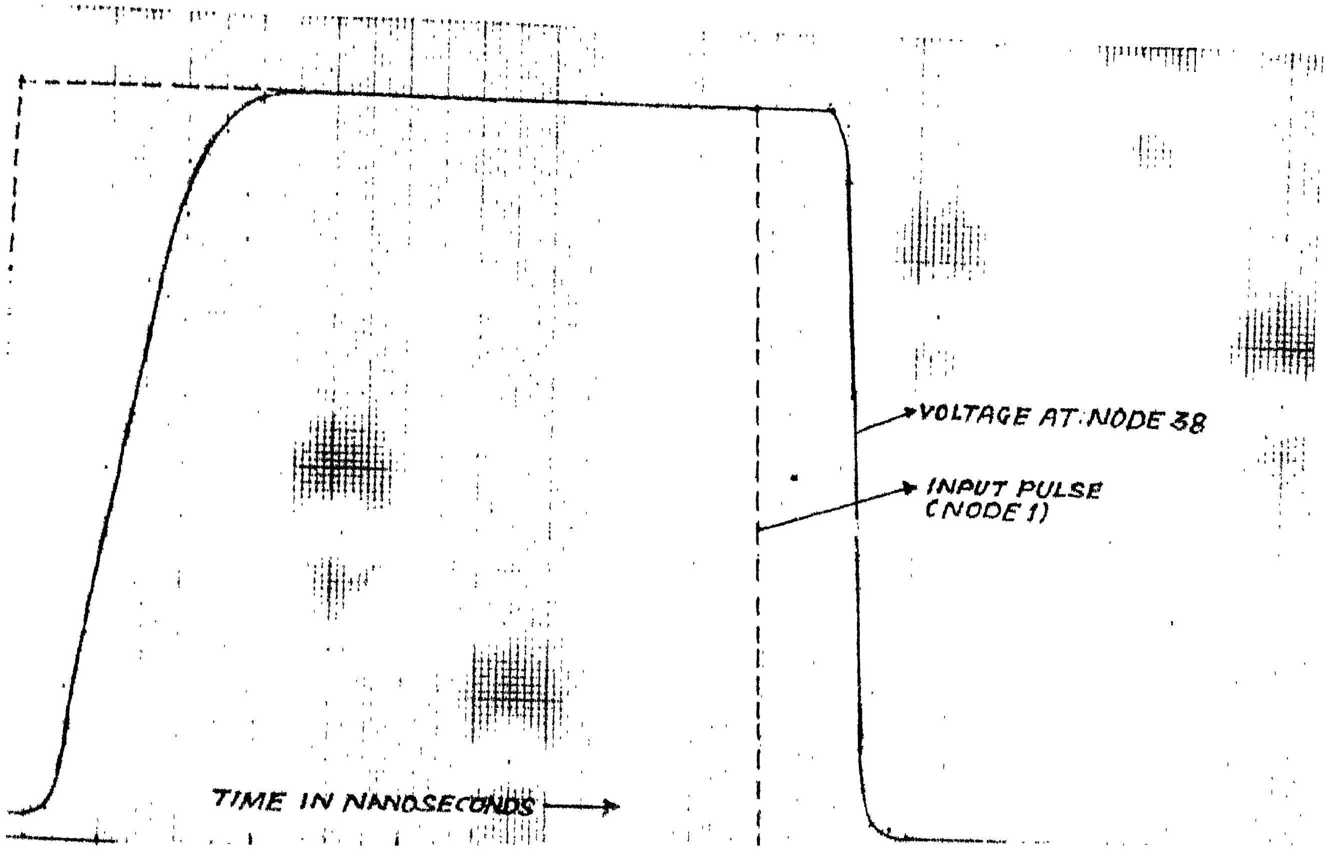

4·0
3·0
2·0
1·0
0·0
INPUT VOLTAGE (NODE 1)
DC TRANSFER CHARACTERISTIC
OUTPUT VOLTAGE
(NODE 38)
0·0
2·0
3·0
4·0

FIG B·2


VOLTAGE AT NODE 38
INPUT PULSE (NODE 1)
TIME IN NANOSECONDS

```
XANDPR2 9 11 6 ANDPR
CA1 17 60 5FF
CA2 17 61 5FF
XANDPR3 9 12 7 ANDPR
XANDPR4 9 13 8 ANDPR
XANDPR5 14 15 12 ANDPR
XANDPR6 14 16 13 ANDPR
XANDPR7 17 18 15 ANDPR
XANDPR8 17 19 16 ANDPR
CA3 26 23 5FF
CA4 27 23 5FF
CA5 20 21 5FF
CA6 20 22 5FF
XANDPR9 14 60 10 ANDPR
XANDPR10 14 61 11 ANDPR
XANDPR11 23 21 24 ANDPR
XANDPR12 23 22 25 ANDPR
XANDPR13 28 24 29 ANDPR
XANDPR14 28 25 30 ANDPR
XANDPR15 33 29 34 ANDPR
XANDPR16 33 30 35 ANDPR
CA7 23 26 5FF
CA8 23 37 5FF
XANDPR17 20 26 18 ANDPR
XANDPR18 20 27 19 ANDPR
XANDPR19 33 31 36 ANDPR
XANDPR20 33 32 37 ANDPR
R1 33 45 600
R2 28 47 600
R3 23 49 600
R4 20 51 600
R5 17 53 600
R6 14 55 600
R7 9  57 600
XORPR1 57 70 58 ORPR
XORPR2 55 70 56 ORPR
XORPR3 53 70 54 ORPR
XORPR4 51 70 52 ORPR
CO1 59 58 5FF
CO2 59 56 5FF
XORPR5 49 42 50 ORPR
XORPR6 47 42 48 ORPR
XORPR7 45 42 46 ORPR
CO3 43 46 5FF
CO4 43 48 5FF
CO5 43 50 5FF
CO6 43 52 5FF
CO7 43 54 5FF
XAD1 6 9 100 ANDDIF
```

```
XAD2 11 14 100 ANDDIF
XAD4 60 17 100 ANDDIF
XAD5 16 17 100 ANDDIF
XAD6 19 20 100 ANDDIF
XAD7 21 20 100 ANDDIF
XAD8 25 23 100 ANDDIF
XAD9 26 23 100 ANDDIF
XAD10 30 28 100 ANDDIF
XAD11 31 28 100 ANDDIF
XAD12 34 33 100 ANDDIF
XAD13 37 33 100 ANDDIF
XNAD1 5 9 NOADIF
XNAD2 7 9 NOADIF
XNAD3 8 9 NOADIF
XNAD4 10 14 NOADIF
XNAD5 13 14 NOADIF
XNAD6 15 17 NOADIF
XNAD7 61 17 NOADIF
XNAD8 22 20 NOADIF
XNAD9 18 20 NOADIF
XNAD10 24 23 NOADIF
XNAD11 27 23 NOADIF
XNAD12 29 28 NOADIF
XNAD13 32 28 NOADIF
XNAD14 35 33 NOADIF
XNAD15 36 33 NOADIF
XOD1 58 70 100 ORDIF
XOD2 56 59 100 ORDIF
XOD3 54 59 100 ORDIF
XOD4 52 42 100 ORDIF
XOD5 50 42 100 ORDIF
XOD6 48 43 100 ORDIF
XOD7 46 43 100 ORDIF
XNOD1 58 59 NORDIF
XNOD2 56 70 NORDIF
XNOD3 54 70 NORDIF
XNOD4 52 43 NORDIF
XNOD5 50 43 NORDIF
XNOD6 48 42 NORDIF
XNOD7 46 42 NORDIF
.SUBCKT IP 2 3 4 10
.SUBCKT NIP1 2 3 4
V1 1 0 DC 5V
C1 2 0 0.5FF
C2 3 0 5.6FF
C3 1 3 0.8FF
ME1 3 2 0 4 MENH L=50 W=10U AS=217P AD=100P
+PS=129U PD=30U
MD1 1 3 3 4 MDEP L=10U W=5U AS=113P AD=63P
+PS=30U PD=20U
```

```
.ENDS IP
.SUBCKT ANDPU 1 3
VE1 2 o 5
C1 1 0 0.72FF
MD2 2 1 1 3 MDEP L=10U W=5U AS=113P AD=113P PS=40U PD=40U
.ENDS ANDPU
.SUBCKT OP 1 2 10
VE1 3 0 5V
C4 4 0 1.92FF
C5 3 5 1.6FF
C6 5 0 2.8FF
C7 1 0 1.1FF
C7D 1 3 1.6FF
R3 1 4 210
R4 5 2 217
ME1 5 4 0 10 MENH
+L=5U W=10U AS=56P AD=56P PS*35U PD=15U
MD1 3 5 5 10 MDEP L=10U W=5U AS=56P AD=112P PS=15U PD=40U
MD2 3 1 1 10 MDEP L=10U W=5U AS=313P AD=113P PS=70U PD=40U
.ENDS OP
.SUBCKT ANDDIF 1 2 3
C1 1 0 1.2FF
C2 2 0 1.6FF
ME1 2 1 0 3 MENH L=5U W=10U AS=225P PS=35U AD=100P PD=20U
.ENDS ANDDIF
.SUBCKT NOADIF 1 2
C1 1 0 1.7FF
C2 2 0 2.0FF
.ENDS NOADIF
.SUBCKT ANDPR 1 2 3
C1 1 3 5.0FF
R1 2 3 210
.ENDS ANDPR
.SUBCKT ORDIF 1 2 3
C1 2 0 1FF
C2 1 0 0.96FF
ME1 2 1 0 3 MENH
+L=5U W=10U AS=225P AD=200P PS=75U PD=20U
.ENDS ORDIF
.SUBCKT NORDIF 1 2
C1 2 0 1.7FF
C2 1 0 1.5FF
.ENDS NORDIF
.SUBCKT ORPR 1 2 3
C1 2 1 0.5FF
R1 1 3 180
.ENDS ORPR
.MODEL MENH NMOS LEVEL=3 TOX=0.3U RSH=35 NSUB=3.5E14 TPG=+1
```

```
+LD=02U UO=700 XJ=0.3U CJ=25U CJSW=150P GAMMA=0.1 CGSD=2.8E-10
+NFS=1E10 THETA=0.1 ETA=0.25 KAPPA=0.5 CGDD=2.8E-10 PB=0.7
+VTO=0.781 VMAX=15E4 UEXP=0.1 UTRA=0.3 LAMBDA=0.02 UCRIT=1.0E4
.MODEL MDEP NMOS LEVEL=3 TOX=0.3U RSH=35 NSUB=35E14 TPG=+1
+LD=0.2U UO=700 XJ=0.3U CJ=25U CJSW=150P GAMMA=0.1 CGSO=2.8E-10
+NFS=1E10 THETA=0.035 ETA=0.25 KAPPA=0.5 CGDO=2.8E-10 PB=0.7
+VTO=-2.2V VMAX=15E4 UEXP=0.1 UTRA=0.3 LAMBDA=0.02 UCRIT=1.0E4
R40 40 0 100MEG
R41 41 0 100MEG
R38 38 0 100MEG
R39 39 0 100MEG
VGR 0 100 3V
VIN2 2 0 OV
VIN3 3 0 OV
VIN4 4 0 OV
VIN1 1 0 PULSE o 5 20N 0 0 400N 1200N
C38 38 0 0.1PF
C39 39 0 0.1PF
C40 40 0 0.1PF
C41 41 0 0.1PF
.DC VIN1 0 5 0.25
.TRAN 10N 1200N
.PRINT DC V(41) V(40) V(38) V(39) V(43) V(42) V(70) V(59)
.PRINT TRAN V(1) V(41) V(40) V(38) V(39)
.OPTIONS PIVTOL=1.0E-14
.END
```

REFERENCES

[1] C. Mead and L. Conway, 'Introduction to VLSI Systems', Addison-Wesley, 1980.

[2] J. Mavor, P.B. Denyer and M.A. Jack, 'Introduction to MOS LSI Design', Addison-Wesley, 1983.

[3] H. Fleisher and L.I. Maissel, 'An Introduction to Array Logic', IBM J. Res.Develop. Jan. 1975, 98-109.

[4] M.S. Schmookler, 'Design of Large ALUs Using Multiple PLA Macros', IBM, J. Res. Develop. January, 1980, 2-15.

[5] R.L. Golden, P.A. Latus and P. Lavy, 'Design Automation and Programmable Logic Array Macro', IBM, J.Rec. Develop. January, 1980, 23-31.

[6] M. Marris Mano, 'Digital Logic and Computer Design', Prentice Hall, 1979.

[7] J. McCluskey, 'Design of Digital Computers', Springer-Verlag, 1975.

[8] Bartee, T.C., 'Computer Design of Multiple Output Logical Networks', IRE, 1961

[9] Z. Kohavi, 'Switching and Finite Automata Theory; 2nd edition, McGraw-Hill.

[10] B. Teel and D. Wilde, 'Logic Minimizer for VLSI PLA Design', 19th Design Automation Conference Proceedings.

[11] V.K. Agarwal, M.R. Dagenais and N.C. Rumin, McBOOLE: A new procedure for exact logic minimization, IEEE Trans. CAD, vol. CAD-5, 1986, 229-238

[12] G.D. Hatchel, A.R. Newton, and A.L. Sangiovanni - Vincentelli, 'An Algorithm for Optimal PLA Folding', IEEE Transaction on CAD of IC, Vol. CAD-1, No.2, April'82, 63-76.

[13] G.D. Hatchel, A.R. Newton, A.L. Sangiovanni - Vincentelli, 'Some Results in Optimal PLA Folding', Proc. Int. Circuits and Computer Conference, Oct, 1980, 1023-1027.

[14] W. Grass, 'A Depth First Branch and Bound Algorithm for Optimal PLA Folding', Proc. 19th Design Automation Conference, Sept. 1982, 133-140.

[15]. J.R. Egan, and C.I. Liu, 'Bipartite Folding and Partitioning of a PLA', IEEE Transaction on CAD, Vol. CAD-3, No.3, July '84, 191-199.

[16] Sun Young Hwang, R.W. Dutton and T. Blank, 'A Best First Search Algorithm for Optimal PLA Folding', IEEE Transaction on CAD, Vol. CAD-5, No.3, July 1986, 433-441.

[17] A.V. Aho, J.E. Hopcroft and J.D. Ullman, 'Data Structures and Algorithms', Addison-Wesley, 1983.

[18] E. Horowitz and S. Sahni, 'Fundamentals of Data Structures in PASCALS', Computer Science Press, 1984.

[19] T.H. Sreenivas, 'Computer Aided Design of Some MOS LSI/VLSI Functional Modules', M.Tech. Thesis, I.I.T. Kanpur, 1986.

[20] L.A. Glasser and D.W. Dobberphul, 'Design and Analysis of VLSI Circuits', Addison-Wesley, 1985.

[21] R.H. Krambeck et al., 'High Speed Compact Circuit with CMOS',IEEE JSSC, Vol. SC-17, No.3 June 1982, 614-619.

[22] N.N. Biswas and B. Gurunath, Bangalore, 'An algorith for the Optimal Minimization of Programmable Logic Array', INT J. Electronics 1986, Vol. 60, No. 6, 709-725.

[23] Y.H. Su and O.L. Dietmeyer, 'Computer Reduction of Two-Level Multiple Output Switching Circuits', IEEE Trans. Comput. Vol.18, Jan. 1980, 58-63.

[24] S.J. Hong et al, MINI : A Heuristic Approach for Logic Minimization, IBM Jl of R and D Vo. 18, 443-458.

[25] D.W. Brown, 'A State Machine Synthesizer-SMS', Proc. of 18th Design Auto. Conf., 301-305.

[26] J.F. Martiner-Graballido and V.M. Powers, PRONTO : Quick PLA Product Reduction', Proc. 20th Design Auto Conf. 545-550.

[27] ayton, R.K. et al, 'ESPRESSO II' A New Logic Minimizer for PLAs, IEEE Trans. on CAD, July 1984, 370-376.

[28] Sasao, T, 'Multiple Values Decomposition of Generalized Boolean Functions and the Complexity of Programmable Logi Arrays, IEEE Tran. Comput. Vol. C-30, No. 9, September 1981, 635-643.